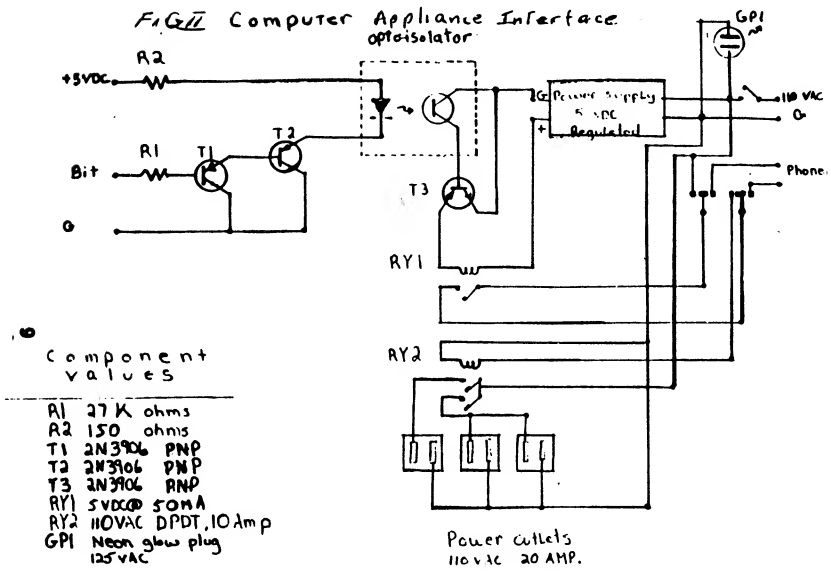
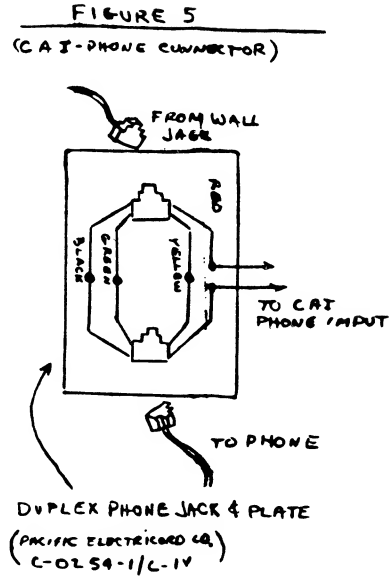
ATARI COMPUTER ENTHUSIASTS
3662 Vine Maple Dr. Eugene OR 97405

MARCH 1985
Editors: Mike Dunn, Jim Bumpas, Larry Gold



Component values

R1	27 K ohms
R2	150 ohms
T1	2N3906 PNP
T2	2N3906 PNP
T3	2N3906 PNP
RY1	5VDC@ 50mA
RY2	110VAC DPDT, 10Amp
GPI	Neon glow plug 125 VAC



CAI INTERFACES

News and Reviews

Mike Dunn, Co-Editor

First of all, please remember our dues are now $14 year or $24 airmail overseas. Groups may order 10 or more at a 40% discount to the same address by 3rd class mail or plus $12 year priority mail.

Last month, a new Atari magazine was started called **The Atari Explorer**. This replaces the old magazine put out by the Atari company, The Atari Connection, and includes all the information about the new Ataris. There is also a very nice article by David and Dorothy Heller about our club which has brought us many new members (Thanks, Dave and Dorothy!). There are many fine articles and you can get yours for only $15 a year (6 issues). I had to wait until I saw it at a newstand to buy it, and was pleased with it. Incidently, the Hellers have introduced ACE to many members through their book, **Free Software for the Atari,** so we have a lot to be thankful to them. We are all anxiously awaiting the arrival of the new Atari Line and hope to give you the scoop on them after we can try them.

This month, we received two new book in the **OMNI** series by Collier Books (division of Macmillan Publishing Co., 866 Third Ave., N.Y. 10022). The first was **OMNI Complete Catalog of Computer Hardware and Accessories** ($14), a goldmine of practical information, advice and reviews on all types of computer equipment (they really like the Atari 800XL!). For someone like myself who goes to computer garage sales and buys slightly used items, it is indispensable. Lots of very useful information, and, of the reviews of equipment I know about, very accurate. The other is the new **OMNI Online Database Directory 1985** ($15), which is a complete evaluation of over 1100 databases including the usual ones such as the Source and CompuServe, and many special service ones ranging in cost from nothing other than the phone bill to $1000 a month or more. Incidently, I hear the **Delphi** is very good and now has a large Atari SIG, and costs $6 hour including 1200 BAUD.

Jerry White, the prolific software author of many well known Atari programs (Trivia Trek, Poker S.A.M., U.S. Adventure, MusicBox, FileIt2, and many others) now has come out with a utility disk of programs he uses commercially in his program development. Called **Animation Using Character Graphics — a tutorial/utility package** it is available only direct from Jerry for $12 (18 Hickory Lane, Levittown, NY 11756. The programs are public domain — you can use them for your program development — but the disk is not. Using your Atari and a program such as the Datasoft MicroPainter, Graphics Master or the Versa Writer, it allows you to make pictures on the screen and convert them to character graphics requiring much less memory, animate them, etc. The disk has seven utilities and is designed for programmers.

Synapse, the originators of FileManager and the Syn series, as well as a number of fine games is now a division of Broderbund. They have just released a number of interactive novels which should be very interesting, to say the least. This is a new art form, and allows you to pick alternate paths so the novel has an almost infinite number of variations.

In future issues, we have many fine programs and articles. **The U.K. Atari Owners Club** (POB 3, Rayleigh, Essex, England, 7lbs. Sterling 4 issues or 10lbs50 Airmail) has a very fine newsletter and many programs of the type Stan Ockers and Sydney Brown give us, and we hope to reprint some of them over the next few months. Paul Freeman, a new ACE member, sent us a sample disk of his programs and they are also of this quality; he is an experienced writer also and has promised us articles to go along with these programs — wait till you see his educational program on the living cell!! Ralph Walden will be submiting programs in ACE-C, and the prolific John Kelly will continue to send mostly educational programs. Not to mention Stan Ockers and Sydney Brown. Lots of good stuff coming up.

****

FLASH

The March issue of BYTE, now on the newstands, feature the new Atari computers, and they like them!!

NEXT MONTH: A wonderful educational program by Paul Freeman, The Living Cell, has impressive graphics and is very entertaining- welcome to ACE, Paul. Stan Ockers will continue his Label program as mentioned above, and who knows what else?

****

BUMPAS REVIEWS

Correction! In the February, 1985 issue, I reviewed the game, **FIELD OF FIRE,** a very good game. Unfortunately I attributed the game to Avalon Hill. Avalon Hill might wish they had the game in their line-up, but the game is sold by STRATEGIC SIMULATIONS, Inc. for $39.95. I hope no one was unable to find the game because of my error.

Atari continues to do things right. Apparently, DOS 3.0 will be treated to a well-deserved exit. Enter DOS 2.5!

DOS 2.5 will be easy to operate for anyone familiar with DOS 2.0. It presents the familiar screen menu. It will freely access any disk formatted in the DOS 2.0 type format. But it has some additions.

It permits automatic use of the dual density format of the Atari 1050 and the Indus GT. Whenever you format a disk, it assumes you want the 1010 sectors available in this mode. If the drive responds, unable to format in this mode, the DOS automatically assumes it's an 810 and formats the disk in single density. A new "P" command is added to the menu to force single density formatting if desired. Appropriate XIO commands have been added to access these functions from BASIC.

DOS 2.5 gives you 1010 sectors (999+ shows on the Directory when the disk is empty). When the disk is full, any files in sectors 721-1010 are highlighted. These files will not be accessible with any other DOS. I tried MYDOS 3.013, SmartDOS, TOPDOS, DOS 2.0, DOS 2.6f, DOSXL and MachDOS. All of them read the disk as though it was single density with only 720 sectors. Even the file names of the files in those higher sectors could not be listed. They are invisible to these DOS.

Sector reading utilities, such as are available in DiskWiz, are able to manipulate the data in these sectors. And SynFile+ and SynCalc are fully able to use files in this format. This is a minor incompatibility of DOS 2.5 with other DOS. Inconsequential when compared with the problems of DOS 3.0. I was unable to use DOS 2.5 to access a double density disk on the Indus GT. Another minor problem. This is the only item I want to see improved.

Just as with double density disks, you cannot read a dual density disk on an 810-type drive, no matter what DOS you use. You must have a drive capable of accessing the 1050 dual mode.

The beta-test disk from Atari had another interesting file on it, which I was unable to use. It has an intriguing name: "RAMDISK.SYS". I believe this is a handler which automatically makes the extra 64k a available to the users of the new 130XE machines! I dunno — maybe it will also work with Axlon and Mosaic boards in our old 800's. If anyone tries this, let us know and we'll share the information.

My drive #1 is an Atari 1050. I use an Indus GT as drive 2. DOS 2.5 provides me with the first meaningful opportunity to use the 1050's dual mode since I chose not to use DOS 3.0. DOS 2.5 is yet another sign that the combination of Tramiel and Atari produces a computer company which knows what to do in today's market place.

The Journal of Computers in Mathematics and Science Teaching is a quarterly published by the Association for Computers in Mathematics and Science Teaching. Subscriptions are $18 to: ACM-ST, Box 4455, Austin, TX 78765.

User-to-User plans to publish a quarterly catalog of "wierdware" for $4 an issue. July, 1985 is the projected first publication date. They plan to have more than 200 programs in this first issue. The programs will all be priced under $20. Rates for program descriptions is $15/100 words. So, if you have a program you developed for your own use, maybe someone else might also have a use for it.

Non-programmers will have the opportunity to buy new and current software at reasonable prices. Clubs can advertise their libraries. The catalog will also offer clubs additional exposure in which to expand their membership base. The catalog will include a special appendix for clubs.

User-to-User is based on the premise that if there was a need for someone to write a particular piece of software, then there are others who can also use it. The catalog will be advertized in national computer magazines. For more information, send a SASE to User-to-User, Box 2605, Eugene, OR 97402.

CMOS 6502

Now the price of CMOS 6502 chips are as low as $8, many Atari users are upgrading to this chip by replacing their old 6502 CPU. There are a number of advantages. The new chip adds a couple dozen new op-codes which make programming easier (these programs will only run on a CMOS chip). The new chip is "low-power", meaning it uses only about 1/100 the power of the old chip — less chance of over-heating (mine never did overheat). The new chip also has a better floating point math package, so it crunches numbers faster than the old chip. Best of all, the CMOS is perfectly compatible with the Atari 400/800.

Now for the bad news. It's not compatible with the XL Ataris, and so probably not with the XE series, either. And even 400/800 users might not want to upgrade because certain software vendors sell protected software which crashes upon finding a non-conforming chip on the board. I believe some of the products from Electronic Arts and Synapse will not work with this chip installed. I could not get SynFile + to work, but SynCalc did.

If you don't use the few products which crash out when they find these chips, you might find them a useful and inexpensive upgrade to make.

COMPUTELY DIFFERENT ($1 and a stamped, self-addressed #10 envelope to Maia Nemzek, 1818 20th Avenue #302, Seattle, WA 98122) intends to stimulate creative ideas, lively discussion, and high-quality applications. The following themes are in the first issue: Noxious Effects of Computer Technology; Computer Applications in the Arts; Disalienation: Computing for Connectedness; Are Computers Political?; Home and Personal Applications: Types of software that don't exist ... yet.

BEYOND CASTLE WOLFENSTEIN

Beyond Castle Wolfensein is a sequel to the adventure game Castle Wolfenstein. The object of this game is to sneak into Hitler's private bunker, find the bomb hidden by your cohorts and place it in the conference room. You are suplied with a gun, some money, and a random number of passes. Not all of the passes supplied to you are the correct passes for that level. More bullets, money and passes can be obtained from the bodies of dead guards (of course you must kill them first).

The guards you encounter may either be bribed or shown a pass. If you get the wrong pass beware. Guards at the main desk to each room complex may be bribed for clues, but these are often cryptic at best. Each room has at least one door or way out. It may also have a locked closet, which you must pick to enter. The game states that by listening carefully you can hear the tumblers click when you get the correct combination. Unfortunately if there is a guard present the sound as he marches back and forth will drown out the tumbler's click.

This game may be played with either a joystick or the keyboard. The use of the joystick creates a much better feel for the fact that you are often required to aim the gun in a rather precise manner to accomplish a task, whether killing a guard or opening a locked closet. There are five levels to the game. With the toughest level simply called "?????". You are allowed all the standerd save game options plus some quick start games such as New Game New Bunker level one or New Game New Bunker current level. Along the way in the game beware of what you leave behind because a dead guard which is discovered on another level can trigger an alarm and then getting out is truly a challenge. If you enjoyed Castle Wolfenstein then you'll enjoy this one as well.

— Nick Chrones

COMPUTER AMBUSH

Computer Ambush (SSI, $60) is a squad-level game representing infantry combat in World War II. You can lead from 1-10 soldiers into battle from North Africa to France, or Italy to Russia.

Your mission takes place in a small town. Your objective can be to mop up some German holdouts, or to destroy a Nazi command post. Your weapons include rifles, automatic rifles, machineguns, grenades, knives, and plastic explosives. Both natural and unlimited sighting are allowed, along with three skill levels: Volksgrenadiers (green), Wehrmacht (average), or SS (excellent).

I always play the SS, and I usually lose! The game disk includes 5 solitare scenarios and 7 two-player scenarios. One of the two-player games is a design-your-own where players can create any situation of WW II they want. Two 10-man squads are included on the disk, and the game has features for the creation of more squads. Each of your soldiers is rated for strength, dexterity, firing ability, and throwing ability. To help those of us who have difficulty handling ten soldiers at a time, SSI thoughtfully included two plastic covered maps, two grease pencils, and two quick reference cards.

I really LOVE this game. It uses a proportional movement, so all the actions of your soldiers can be simultaneous. For those of you who like board games such as Squad Leader by Avalon Hill, and are thoroughly disgusted at extensive line-of-sight rules, rejoice! In Computer Ambush, the computer handles the hard parts and lets you worry about simple things like: Is that soldier's rifle loaded? or Will the grenade blow up your soldier as well as the Germans? This is as much a role playing game as it is a WW II simulation. An excellent game for the advanced wargamer

— Aaron Ness

VP's RAMBLINGS

I've heard from a number of you expressing your complaints about magazines, software houses etc., and when we have what we consider to be enough complaints about a specific subject or business we will try to act on your behalf. Remember if you don't write we can't right your wrong.

We are going to have a contest, and this contest is for those of you who like to do graphics. We want new designs for the cover of the newsletter, both the overall format, and specific graphic dumps. In addition we want a new opening for our BBS. The prizes will be free memberships in ACE, special levels on the BBS no one else has, and disks of programs made up with customized menus just for the winners. Mext month I will lay down the rules and everyone is eligible whether they are a member or not. There will be many winners and I hope to see alot of good ideas and graphics from all of you who read ACE.

If any of you want to see specific types of programs, articles, etc., there again please let us know so we may make this newsletter more responsive to your needs. If we don't hear from you we don't always know what you want and can only guess and put in the material we think you want to read. We also don't always get the material we want to see so if you have programs or articles you want to share with other Atari users let us know about them and if we feel they will go in this newsletter we will publish what you send. Somewhere someone may be looking for that program you wrote and haven't let anyone know about.

— Larry Gold

MMG BASIC COMPILER

This product has been well advertised in **ANTIC** and in **ANALOG** with full page ads proclaiming it as the "ultimate" Basic compiler. From my experience with it I find it is does not live up to its claims. It is, in fact, not a very reliable compiler and certainly not worth the $99 retail price tag.

I tried this compiler for over 2 months during which time I used it on 41 different Atari Basic programs. Of the total number of programs, only 17 compiled successfully.

My procedure was to first testout the basic program to make sure it functioned before compiling. Then as a precaution, I cleared the variable table thru the List, New, Enter, Save technique. My next step was to determine whether the Floating Point Library was necessary and load the appropriate files from the Master Disk into a disk in Drive 2. At this time I also loaded the program I wished to compile onto Drive 2. I used 2 drives to avoid extensive disk swapping.

Unless there were obvious decimals and fractions I used the Integer library file and compiled the program. During compilation I seldom encountered system errors, but when I did it was usually when I had attemped to chain several programs together (supposedly allowable). At this time I often got a 129 error — too many channels open (Even when they were closed). I got around this by trapping the error to the line running the next program in the chain.

Another system error I got a lot was "Undefined Line number" which means the program referenced a non-existent line. Since this was never the case, I bypassed the error and finished the compilation anyway. Somtimes the program ran anyway.

Sometimes a program failed to run with no error messages or explanation of any kind from the program or manual. This frustrating development resulted in (1 System lockup or (2 a blown display list — or both.

My last procedure was to libraries — Integer for Floating Point or Vice Versa. This only worked once.

It should be obvious by now the **MMG COMPILER** is not the promised answer. Save your money! If you need a compiler stick with the ABC MONARCH, for the present, as the most reliable compiler even with its "Floating Point" limitations.

— Graham Smith

HAPPY DRIVES BASIC SLOWDOWN

Hello folks. I realize it has been a LONG time since I last sat down and wrote an article. Well, here is a neat program I wrote a while ago. This program allows an owner of HAPPY DRIVES to perform a 'SLOWDOWN' in BASIC. I have included many REM statements and used the Atari labels as variable names to make understanding the program easier.

— Shane Rolin

INDEXED FILES MADE EASY

Indexed files can be used for quick, random retrieval of data, much faster than for direct or sequential files. Generally speaking, indexed files consist of two files. One file contains the data you require and the other file contains the indices or keys pointing to the data records. In this article, as a demonstration, we will create a quick-access telephone number file.

By carefully studying how to use the BASIC commands, NOTE and POINT, we can easily see how to create and use indexed files. Briefly, NOTE #X,SEC,BYT will determine the position of the disk drive's pointer on the file opened on channel #X. It will return the values of the sector and byte of the pointer to the variables, which in this case are SEC and BYT. POINT #X,SEC,BYT will set the pointer to the byte located at sector, SEC, and byte, BYT, on the file opened to channel #X.

In Listing 1, which is the main program, the keys or indices are held in a file called "KEYS". During execution the entire file is in memory allowing rapid access. Each entry in "KEYS" is 12 bytes long: the first six bytes are the key and the next six contain the sector and byte information pointing to the corresponding telephone data stored on disk in another file called "TELEPHON.DAT".

In order to easily remember the key, I use the first three letters of the last name and the first three letters of the first name. Thus the key for John Smith is SMIJOH. My "TELEPHON.DAT"file has the same structure as one published by Jerry White ("Phone Book", ANTIC, Feb. 1984, Vol. 2, No. 11). If you already have used this program and have a phone book file, Listing 2 will create a "KEYS" file from your "TELEPHON.DAT" file. If you have a name such as John Smiley which will duplicate another key, such as John Smith, one of them will be given the key SMIJO1. If there is a triplicate, it will be SMIJO2 and so on.

When you load and run the main program, it will check if your data files are on the disk in drive one. If you have a data file on your disk, the options menu will be the first screen to appear. If the file is not present or if you are entering your first data record, you will be prompted and asked if you want to continue. You will then see the ADD screen. Just follow the instructions and type in your key and the corresponding data. When you are finished adding data, type "END". You will then see the menu of options including: ADD, CHANGE (can be used to change any part of a record), DELETE, INQUIRY/PRINT (to view or make a hard copy of a particular record), SCAN RECORDS (to sequentially view the records and remind yourself of the keys in case you have forgotten or haven't printed a copy), and END to return to BASIC.

When a key is typed into the computer, the data are retrieved from the disk and appear on the screen in the order of a second or two even with hundreds of records. This process is expedited by a machine language string search which locates the position of the key in the string, S$, which contains all the keys and their corresponding variables SEC and BYT (that is, sector and byte). A POINT #1,SEC,BYT statement then locates the telephone data on the disk.

When a record is deleted, the program inserts six blanks for the deleted key in the string S$. By searching for blanks, the program can write a new record over a deleted one thus saving disk space and memory. However, until a deleted record is overwritten, it can still be accessed through SCAN RECORDS.

Another feature worth noting in the program is the use of control characters in the strings SEL$ and FIELD$ defined in lines 190 and 200 respectively. When SEL$ is printed on the screen, the menu of options appears. And FIELD$ will print the template for the data entry screen. This approach saves many lines of code. Due to an absence of an end-of-line character (i.e. RETURN), I had to use (ESCAPE DOWN) followed by (ESCAPE SHIFT-DELETE) in these strings. This combination of keys achieves almost the same effect as a RETURN.

This indexed file is probably the simplest one possible. The next level of complexity might include a sort subroutine to put the keys in alphabetical order. Then the data could be accessed in alphabetical order based on the keys. Since there are several excellent machine language sort routines in print, I leave the next step as an exercise for the reader.

List of Variables

A	Index used to identify key as substring in S$
AREA$	Area code
BLK$	String of blank characters
BYT	Value of byte where record is stored on disk, used with SEC
BYT$	String variable of BYT, STR$(BYT)
DASH$	String of dashes
DAT$	"D:TELEPHON.DAT", data file
EXCH$	Telephone exchange
FIELD$	String containing information to print data entry screen.
FIRST$	First name
FLAG	= 0: append new record to end of file = 1: file TELEPHON.DAT doesn't exist. = 2: replace deleted record with new record.
FX	= 0: field has variable length = 1: field has fixed length
K$	"D:KEYS", index file
KEY$	key or index
KTEMP$	temporary string
L( )	array for length of each field
LAST$	Last name
LINE	Statement number for continuation after error
LINEY	Statement number for continuation after error
M1$	Message string
ME$	Message string
ML$	Machine language string
PNUM$	Phone number
R( )	Row where field appears on data entry sccreen
REC	Number of records (actually twice the number)
REC$	String containing a single record
S$	String containing keys and pointers to sector and byte.
SEC	Sector where record is stored on disk.
SEC$	String containing value of SEC, STR$(SEC)
SEL$	Contains information to print option menu on screen
ST	Variable used in input and output of S$
TEMP$	Temporary string
X$	used for input of data
Y	Variable used in input and output of S$

OUTLINE OF PROGRAM

10-240	Initialization
1000-1050	Options Menu
2000-2130	ADD option
3000-3120	CHANGE option
4000-4100	DELETE option
5000-5050	INQUIRY/PRINT option
6000-6010	END option
6500-6610	SCAN RECORDS option
7000-7050	Subroutine to read indices from disk
7100-7150	Subroutine to write indices to disk
7500-7580	General data entry subroutine
8000-8100	Subroutine to write data to disk, if they are correct
8200-8260	Subroutine to make hard copy of data on printer
8300-8320	Subroutine to fill data field with trailing blanks
8500-8520	Subroutine to print message on screen
9000-9100	String search subroutine to locate key and print data on screen
10000-10030	Error trapping routine
12000-12550	String-search machine language subroutine

— Gary Wick

Inside the Atari 810

Part 1. The Atari 810 Disk Drive

The Atari 810 drive was an excellent product for the young Atari company back in the early 80's. It quickly became popular until late 1983. The Atari 810 is basicly a Tandon single-sided, single-density raw drive with custom Atari hardware.

The hardware in the 810 is built around the 6507. Yes, a stripped down version of the 6502 microprocessor. The electronics include a 1771 controlled chip, 128 byte Ram chip; a 2316 2k Rom chip. This chip and the Ram chip are replaced when you have HAPPY DRIVES. There is also a 2332 Ram chip. This chip does a job similar to the PIA chip in the Atari Computer, and adds another 128 bytes of memory to the drive.

The Atari 810 is accessed from the computer serially at 19,200 baud. This simply means the 810 is about 20 times slower than the Apple drives. The major difference between the Atari and Apple drives is the Atari and 810 communicate by high level commands and pass data back and forth. The Apple 6502 controls the Apple drive, therefore through software, the Apple drives can be totally controlled.

On the Atari, there are a total of 5 commands by which the computer and drive communicate. These commands are:

1. Get Sector number. The computer sends this command to the drive along with a sector number from $1-$2D0 and for Happy Drives $800-$13FF. The Drive returns 128 bytes of data.

2. Put Sector number. The computer sends this command to the drive along with a sector number and 128 bytes of data. The sector number is between $1-$2D0 and for Happy Drives $800-$13FF. The drive writes 128 bytes of data.

3. Put Sector number with Verify. The computer sends this command along with the sector number and 128 bytes of data. The drive writes the data to the appropriate sector, reads the sector, and sends the data back to the computer. The computer checks both sets of data to make sure that they are the same.

4. Format disk. Upon recieving this command, the drive formats the disk in a 40 track, 18 sector per track, 128 byte per sector format.

5. Drive Status. Upon recieving this command, the drive sends it's status to the computer.

More Soon . . . .

— Shane Rolin

INDEX FILES BY GERRY WICK

```
    10 REM ******************************
20 REM **                            **
22 REM **      INDEXSETUP            **
24 REM **                            **
26 REM **          by                **
28 REM **      GERRY WICK            **
30 REM **                            **
32 REM ** ACE Newsletter Mar. 85     **
34 REM **  3662 Vine Maple Dr.       **
36 REM **  Eugene, Or. 97405         **
38 REM **      $14 year              **
40 REM ******************************
42 REM
50 GOSUB 12000
100 DIM REC$(34),KEY$(6),S$(6000),SEC$
(3),BYT$(3),X$(3)
1000 ? "%INSERT TELEPHON.DAT DISK AND"
:? :? "PRESS RETURN WHEN READY"
1010 TRAP 1000:INPUT X$
1020 CLOSE #1:OPEN #1,12,0,"D:TELEPHON
.DAT":TRAP 2000
1030 NOTE #1,SEC,BYT:INPUT #1,REC$
1040 K=0
1050 SEC$=STR$(SEC):BYT$=STR$(BYT)
1060 IF LEN(SEC$)<3 THEN X$=SEC$:SEC$(
2)=X$:SEC$(1,1)="0":GOTO 1060
1070 IF LEN(BYT$)<3 THEN X$=BYT$:BYT$(
2)=X$:BYT$(1,1)="0":GOTO 1070
1080 KEY$(1,3)=REC$(1,3):KEY$(4,6)=REC
$(13,15)
1090 REC=LEN(S$)/6:IF REC=0 THEN 1200
1100 A=USR(ADR(ML$),REC,ADR(S$),ADR(KE
Y$))
1110 IF A=0 THEN 1200
1120 K=K+1:KEY$(6,6)=STR$(K):GOTO 1100
1200 S$(LEN(S$)+1)=KEY$:S$(LEN(S$)+1)=
SEC$:S$(LEN(S$)+1)=BYT$
1210 GOTO 1030
2000 GOSUB 7100
2100 END
7100 REM WRITE INDEX TO DISK
7110 CLOSE #1:OPEN #1,8,0,"D:KEY$"
7120 Y=LEN(S$):J=INT(Y/125):? #1;Y
7125 IF Y<125 THEN 7140
7130 FOR I=0 TO J-1:ST=I*125+1:? #1;S$
(ST,ST+124):NEXT I
7140 IF Y/125<>INT(Y/125) THEN ? #1;S$
(J*125+1,Y)
7150 CLOSE #1:REC=LEN(S$)/6:RETURN
12000 RESTORE 12500:DIM ML$(60)
12010 FOR ML=1 TO 60:READ A:ML$(ML,ML)
=CHR$(A):SOUND 0,ML,10,8:NEXT ML:SOUND
 0,0,0,0
12020 RETURN
12500 DATA 104,104,133,213,104,133,212
,104,133,206
12510 DATA 104,133,205,104,133,204,104
,133,203,160
12520 DATA 0,177,203,209,205,208,6,208
,192,6
12530 DATA 208,245,96,165,205,24,105,6
,133,205
12540 DATA 144,2,230,206,165,212,208,6
,165,213
12550 DATA 240,7,198,213,198,212,24,14
4,216,96



10 REM ******************************
20 REM **      INDXFILE BY           **
22 REM **                            **
24 REM **      GERRY  WICK           **
26 REM **                            **
28 REM ** ACE Newsletter Mar. 85     **
30 REM **  3662 Vinemaple Dr.        **
32 REM **  Eugene. Or. 97405         **
34 REM **      $14 year              **
36 REM ******************************
40 REM
50 GOSUB 12000
90 REM SETUP FOR 500 RECORDS
100 DIM BLK$(12),REC$(34),TEMP$(34),LA
ST$(12),FIRST$(12),AREA$(3),EXCH$(3),P
NUM$(4),KEY$(6),S$(6000),DASH$(12)
110 DIM FIELD$(80),SEL$(99),M1$(25),X$
(12),K$(6),DAT$(14),R(6),L(6),ME$(65),
KTEMP$(125),SEC$(3),BYT$(3)
120 K$="D:KEY$":DAT$="D:TELEPHON.DAT":
FLAG=0
150 BLK$(1)=" ":BLK$(12)=" ":BLK$(2)=B
LK$
160 DASH$(1)="-":DASH$(12)="-":DASH$(2
)=DASH$
170 R(0)=5:FOR I=1 TO 5:R(I)=6+I:NEXT
I
180 L(0)=6:L(1)=12:L(2)=12:L(3)=3:L(4)
=3:L(5)=4
190 SEL$="1.ADD     2.CHANGE   3.DELET
E↓4.INQUIRY/PRINT    5.SCAN RECORDS↓
6.RETURN TO BASIC"
200 FIELD$="))<<<KEY↓↓1.LAST NAME↓2.
FIRST NAME↓3.AREA CODE↓4.EXCHANGE↓5
.PHONE NUMBER"
210 M1$="ENTER OPTION "
220 REC=1
230 TRAP 10000:? "%":LINE=120:LINEY=20
00:ME$="THERE ARE NO RECORDS ON THIS D
ISK↓           CONTINUE (Y/N)"
240 GOSUB 7000:TRAP 40000
1000 REM MENU
1010 ? "%":POSITION 15,2:? "MENU":? :?
 :? FIELD$
1020 POSITION 13,15:? "OPTIONS":? :? S
EL$:?
1025 POSITION 2,21:? M1$;BLK$(1,2);"<<
";
1030 INPUT X$:IF ASC(X$)<49 OR ASC(X$)
>54 THEN 1025
1050 ON VAL(X$) GOSUB 2000,3000,4000,5
000,6500,6000
2000 REM ADD
2010 TRAP 40000:? "%":POSITION 15,1:?
"ADD":? :? "     ENTER DATA OR 'END'":
? :? FIELD$
2030 R=R(0):L=L(0):FX=1:A=1:GOSUB 7500
:KEY$=X$:LINE=2000
2040 IF FLAG<>1 THEN A=USR(ADR(ML$),RE
C,ADR(S$),ADR(KEY$)):IF A<>0 THEN ME$=
"DUPLICATE ID":GOSUB 8500:GOTO 2000
2050 IF FLAG<>1 THEN KTEMP$="      ":A
=USR(ADR(ML$),REC,ADR(S$),ADR(KTEMP$))
:IF A=0 THEN FLAG=0:GOTO 2057
2055 IF FLAG<>1 AND A<>0 THEN A=6*(REC
-A)+1:SEC=VAL(S$(A+6,A+8)):BYT=VAL(S$(
A+9,A+11)):FLAG=2:GOTO 2060
2057 A=6*(REC-A)+1
2060 R=R(1):L=L(1):FX=0:GOSUB 7500:GOS
UB 8300:LAST$=X$:REC$=LAST$
2070 R=R(2):L=L(2):FX=0:GOSUB 7500:GOS
UB 8300:FIRST$=X$:REC$(13)=X$
2080 R=R(3):L=L(3):FX=1:GOSUB 7500:ARE
A$=X$:REC$(25)=X$
2090 R=R(4):L=L(4):FX=1:GOSUB 7500:EXC
H$=X$:REC$(28)=X$
2100 R=R(5):L=L(5):FX=1:GOSUB 7500:PNU
M$=X$:REC$(31)=X$
2110 GOSUB 8000:REM ARE THE DATA CORRE
CT?
2120 GOSUB 8200:REM HARD COPY?
2130 GOTO 2000
3000 REM CHANGE
3010 ? "%":POSITION 14,1:? "CHANGE":?
:? "     ENTER DATA OR 'END'":? :? FIE
LD$
3020 R=R(0):L=L(0):FX=1:FLAG=2:GOSUB 7
500:KEY$=X$:LINE=3000:GOSUB 9000:REM S
TRING SEARCH AND PRINT DATA
3030 POSITION 2,15:? "↑ENTER FIELD NU
MBER TO CHANGE":? "(ENTER RETURN TO EN
D)";:INPUT X$
3035 IF X$="" THEN GOTO 3100
3040 IF ASC(X$)<49 OR ASC(X$))53 THEN
```

```
3030
3050 IF VAL(X$)=1 THEN R=R(1):L=L(1):F
X=0:GOSUB 7500:REC$(1,12)=X$:GOTO 3030
3060 IF VAL(X$)=2 THEN R=R(2):L=L(2):F
X=0:GOSUB 7500:REC$(13,24)=X$:GOTO 303
0
3070 IF VAL(X$)=3 THEN R=R(3):L=L(3):F
X=1:GOSUB 7500:REC$(25,27)=X$:GOTO 303
0
3080 IF VAL(X$)=4 THEN R=R(4):L=L(4):F
X=1:GOSUB 7500:REC$(28,30)=X$:GOTO 303
0
3090 IF VAL(X$)=5 THEN R=R(5):L=L(5):F
X=1:GOSUB 7500:REC$(31,34)=X$:GOTO 303
0
3100 GOSUB 8000:REM ARE DATA CORRECT?
3110 GOSUB 8200:REM HARD COPY?
3120 GOTO 3000
4000 REM DELETE
4010 ? "K":POSITION 14,1:? "DELETE":?
:? "     ENTER DATA OR 'END'":? :? FIE
LD$
4020 R=R(0):L=L(0):FX=1:FLAG=2:GOSUB 7
500:KEY$=X$:LINE=4000:GOSUB 9000:R
```

STRING MAGIC by Charlie Parker

```
10 REM EXAMPLE #3 - STRING MAGIC FOR
PAGE FLIPPING.  CHARLIE PARKER, STAR-
FLEET, 7685 S. DATURA, LITTLETON, CO
50 DIM DM$(1):REM CREATE STRING FOR
DISPLAY MEMORY
70 DIM SM1$(480),SM2$(480),SM3$(480):R
EM CREATE STRINGS TO SAVE DISPLAY MEM.
90 GRAPHICS 23:REM USE GR.7 WITH NO WI
NDOW
100 SETCOLOR 4,10,5
110 SETCOLOR 0,0,0
120 POKE 559,0:REM TURN OFF SCREEN
130 REM SET UP PARMS FOR STRING MAGIC
140 REM SET ADDR TO DISPLAY MEMORY
150 ADDR=PEEK(88)+PEEK(89)*256
160 SIZE=3840:REM GR.23 SCREEN SIZE
170 VNUM=0:REM USE 1ST VAR. (DM$)
180 REM CALL THE STRING MAGIC ROUTINE
190 GOSUB 810
200 P=0
210 REM DRAW FIRST VERSION OF BAT
220 COLOR 1
230 PLOT P+5,1
240 DRAWTO P+19,4:DRAWTO P+34,1
250 GOSUB 660
260 IF P<120 THEN P=P+40:GOTO 230
270 REM SAVE DISPLAY MEMORY IN SM1$
280 SM1$=DM$
290 REM CLEAR DISPLAY MEMORY
300 DM$(2)=DM$(1)
310 REM DRAW SECOND VERSION OF BAT
320 P=0
330 PLOT P+5,4
340 DRAWTO P+34,4
350 GOSUB 660
360 IF P<120 THEN P=P+40:GOTO 330
370 REM SAVE DISPLAY MEMORY IN SM2$
380 SM2$=DM$
390 REM CLEAR DISPLAY MEMORY
400 DM$(2)=DM$(1)
410 REM DRAW THIRD VERSION OF BAT
420 P=0
430 PLOT P+5,7
440 DRAWTO P+19,4:DRAWTO P+34,7
450 GOSUB 660
460 IF P<120 THEN P=P+40:GOTO 430
470 REM SAVE DISPLAY MEMORY IN SM3$
480 SM3$=DM$
490 C=0
500 POKE 559,34:REM TURN ON SCREEN
510 REM FLIP THRU THE DIFF VERSIONS
520 DM$(1,480)=SM1$
530 DM$(481,960)=SM2$
540 GOSUB 750
550 DM$(1,480)=SM2$
560 DM$(481,960)=SM3$
570 GOSUB 750
580 DM$(1,480)=SM3$
590 DM$(481,960)=SM2$
600 GOSUB 750
610 DM$(1,480)=SM2$
620 DM$(481,960)=SM1$
630 GOSUB 750
631 IF PEEK(764)<>255 THEN POKE 764,25
5:RUN "D:MENU"
640 GOTO 520
650 REM COMMON ROUTINE TO DRAW HEAD
660 PLOT P+17,3:DRAWTO P+21,3
670 PLOT P+17,5:DRAWTO P+21,5
680 PLOT P+17,2:DRAWTO P+21,2
690 PLOT P+18,6:DRAWTO P+20,6
700 PLOT P+17,1:DRAWTO P+21,1
710 COLOR 3:PLOT P+18,4:PLOT P+20,4
720 COLOR 1
730 RETURN
740 REM DUPLICATE TOP PART OF SCREEN
750 DM$(961,1920)=DM$(1,960)
760 DM$(1921)=DM$(1,1919)
770 REM CHANGE COLOR OF EYES
780 C=C+2:IF C>255 THEN C=12
790 POKE 710,C:RETURN
800 REM STRING MAGIC ROUTINE
810 A=(ADDR-(PEEK(140)+PEEK(141)*256))
:AH=INT(A/256):AL=A-(AH*256):VVT=PEEK(
134)+PEEK(135)*256:Q=VVT+VNUM*8+2
820 IF A<0 THEN A=PEEK(140)+PEEK(141)*
256:GRAPHICS 0:? "CANNOT GO BELOW ADDR
ESS ";A:STOP
830 IF PEEK(Q-2)<128 THEN GRAPHICS 0:?
 "VARIABLE  #";VNUM;" NOT A STRING":ST
OP
840 POKE Q,AL:POKE Q+1,AH:AH=ING(SIZE/
256):AL=SIZE-(AH*256):POKE Q+2,AL:POKE
 Q+3,AH:POKE Q+4,AL:POKE Q+5,AH:RETURN
```

RAMTALKER

```
10 REM RAMTALKER Version 3.0 11/84 A.R
. Holmes --- reprinted from the 1/85 S
tatus Newsletter, Norfolk, VA
20 GRAPHICS 0:POKE 752,1:? :? :? :? "I
nitializing...please wait"
30 FOR I=0 TO 243:READ Z:POKE 1536+I,Z
:NEXT I
40 GOTO 50
50 DIM Z(255),FN$(13):OPEN #1,4,0,"K:"
60 GRAPHICS 2:SETCOLOR 2,0,0:TRAP 60
70 ? #6;" RAMTALKER":? #6
80 ? #6;" 1 record ":? #6;" 2 playback
 ":? #6;" 3 throughput"
90 ? #6;" 4 save":? #6;" 5 load"
100 ? #6;" 6 waveform graph"
110 TRAP 110:GET #1,AN5:IF AN5>54 OR A
N5<49 THEN 110
120 IF AN5>51 THEN 140
130 TRAP 60:POKE 752,1:? "What Sample
Speed";:INPUT S5:IF S5>255 THEN 130
140 ON VAL(CHR$(AN5)) GOTO 160,200,240
,270,330,640
150 REM TALK
160 POKE 208,1:POKE 205,0:POKE 206,64:
POKE 207,55:POKE 209,128
170 A=USR(1536):POKE 562,3:POKE 53775,
3
180 GOTO 60
190 REM PLAYBACK
200 POKE 207,55:POKE 203,0:POKE 204,64
:POKE 208,0:POKE 206,128
210 A=USR(1536):POKE 562,3:POKE 53775,
3
220 GOTO 60
230 REM THROUGHPUT
240 POKE 208,2:POKE 205,0:POKE 206,64:
POKE 207,55:POKE 209,128
250 A=USR(1536):GOTO 240
260 REM SAVE SOUND FILE
270 TRAP 270:POKE 752,1:? "Give file n
ame";:INPUT FN$:IF FN$="" THEN 60
280 IO=4:OPEN #4,8,0,FN$
290 ADDRESS=16384:NUMBER=16383:PROC=11
300 GOSUB 510
310 GOTO 60
320 REM LOAD SOUND FILE
330 TRAP 330:POKE 752,1:? "Give file n
ame";:INPUT FN$:IF FN$="" THEN 60
340 IO=4:OPEN #4,4,0,FN$
350 ADDRESS=16384:NUMBER=16383:PROC=7
360 GOSUB 510
370 GOTO 60
380 DATA 104,169,8,141,31,208,173,31,2
08,41,1,208,249,160,255,162,255,32,149
,6
```

```
390 DATA 136,208,248,169,8,141,31,208,
166,208,224,0,208,3,76,181,6,169,0,141
400 DATA 0,212,141,14,212,141,10,212,1
41,10,212,166,207,32,149,6,173,4,210,1
62
410 DATA 19,142,15,210,162,23,142,10,2
12,142,15,210,142,11,210,174,243,6,224
,0
420 DATA 208,22,41,240,141,242,6,106,1
06,106,106,41,15,9,16,141,1,210,238,24
3
430 DATA 6,76,45,6,106,106,106,106,41,
15,9,16,141,1,210,41,15,13,242,6
440 DATA 206,243,6,160,0,145,205,173,3
1,208,41,1,240,19,230,205,208,163,230,
206
450 DATA 166,206,228,209,208,155,76,15
3,6,202,208,253,96,165,208,201,2,208,1
1,169
460 DATA 0,133,205,169,64,133,206,76,3
7,6,169,64,141,14,212,169,34,141,0,212
470 DATA 96,169,0,141,14,212,141,0,212
,166,207,32,149,6,160,0,177,203,170,10
6
480 DATA 106,106,106,41,15,9,16,141,1,
210,138,41,15,9,16,24,24,24,24,166
490 DATA 207,32,149,6,141,1,210,230,20
3,208,206,230,204,166,204,228,206,208,
206,76
500 DATA 153,6,0,0
510 REM CIO READ/WRITE
520 IO=16*IO
530 IOCB=832+IO:POKE IOCB+2,PROC
540 ADRHI=INT(ADDRESS/256)
550 ADRLO=ADDRESS-ADRHI*256
560 POKE IOCB+4,ADRLO:POKE IOCB+5,ADRH
I
570 NUMHI=INT(NUMBER/256)
580 NUMLO=NUMBER-256*NUMHI
590 POKE IOCB+8,NUMLO:POKE IOCB+9,NUMH
I
600 I=USR(ADR("hhh[ESC]LV[ESC]"),IO)
605 REM Line 600 has inverse "*" after
 the 3 lower case "h"; and inverse "d"
 after the upper case "V"
610 CLOSE #IO/16
620 RETURN
630 REM PLOT WAVEFORM
640 GRAPHICS 8:SETCOLOR 2,0,0:COLOR 1:
POKE 752,1:? "[ESC]During plot, press any
key to          return to main menu"
650 FOR I=1 TO 400:NEXT I
660 GRAPHICS 8+16:SETCOLOR 2,0,0:COLOR
 1:A=0:B=0:C=0:D=0:H=0
670 FOR I=1 TO 4095:POKE 764,255
680 A=PEEK(I+16384)
690 B=PEEK(I+20480)
700 C=PEEK(I+24575)
710 D=PEEK(I+28670)
720 H=H+0.07773:PLOT H,(A/4)-15
740 PLOT H,(B/4)+30
750 PLOT H,(C/4)+75
760 PLOT H,(D/4)+120
770 IF PEEK(764)=255 THEN NEXT I
780 IF PEEK(764)=255 THEN 780
790 GOTO 60
```

****

New Disks

Ready in the next few weeks:

ACE Business Disk #1

This disk will consist of LABLES and, when ready, LIST PRINTING by Stan Ockers, CASH FLOW and INDEXED FILES by Gerry Wick, and DIF CONVERSION by David Fuller. Side 2 will have HOME FINANCIAL DATABASE a shareware disk by Richard Kalagher.

ACE TeleCommunication Disk

We now have programs to allow the Atari 1030 Modem to have upload/download and other intelligent functions, the legendary R.BIN file (see ACE NOV '84 article by Tom Neitzel) that allows you to run most modem programs with the Atari 835/1030 Modems, and the fabled KERMIT, written in ACTION!, de-bugged by MicroBits, in source and in run-time versions for the MicroBits Modems, and maybe other modems as well.

Menus

A collection of Menu programs from various sources collected by one of our members.

All of the above should be available soon, at the usual price of $10 each or $15 double-sided.

snakes cont

```
5480 IF MAN=65 THEN MAN=85:CHECK=1:GOT
0 5700
5490 IF MAN=79 THEN MAN=100:CHECK=1:GO
TO 5700
5500 IF MAN=100 THEN 8000
5550 GOTO 2500
5600 REM SNAKE SOUND
5610 RESTORE 5680
5620 FOR I=1 TO 16:READ TONE
5630 SOUND 0,TONE,10,8:FOR ZZ=1 TO 1:N
EXT ZZ
5640 NEXT I
5650 SOUND 0,0,0,0:FOR ZZ=1 TO 10:NEXT
 ZZ
5660 SOUND 0,243,12,15:FOR ZZ=1 TO 5:N
EXT ZZ
5670 SOUND 0,0,0,0
5680 DATA 40,45,50,57,64,72,81,91,102,
114,128,144,162,182,204,230
5690 GOTO 4400
5700 REM LADDER SOUND
5710 FOR I=1 TO 10
5720 SOUND 0,29,8,15
5730 FOR ZZ=1 TO 1:NEXT ZZ
5740 SOUND 0,0,0,0
5750 FOR ZZ=1 TO 3:NEXT ZZ
5760 NEXT I
5770 GOTO 4400
7000 REM THE COMPUTER HAS WON
7010 FOR I=1 TO 15
7020 SOUND 0,91,10,10:FOR ZZ=1 TO 10:N
EXT ZZ
7030 SOUND 0,0,0,0
7040 NEXT I
7050 FOR I=15 TO 0 STEP -1
7060 SOUND 0,91,10,I:FOR ZZ=1 TO 10:NE
XT ZZ
7070 NEXT I
7999 GOTO 9999
8000 REM THE MAN HAS WON
8010 FOR I=1 TO 15
8020 SOUND 0,71,10,10:FOR ZZ=1 TO 10:N
EXT ZZ
8030 SOUND 0,0,0,0
8040 NEXT I
8050 FOR I=15 TO 0 STEP -1
8060 SOUND 0,71,10,I:FOR ZZ=1 TO 10:NE
XT ZZ
8070 NEXT I
8999 GOTO 9999
9999 FOR ZZ=1 TO 1000:NEXT ZZ:GOTO 2
```

LABELS

```
10 REM ****************************
12 REM ***       LABELS          ***
14 REM ***   S. O.  FEB. 85      ***
16 REM ***   ACE NEWSLETTER      ***
18 REM *** 3662 Vine Maple Dr.   ***
20 REM ***  Eugene, OR 97405     ***
22 REM ***  Mar. 85 $14 year     ***
24 REM ****************************
26 REM
29 REM *** MAIN LOOP ***
30 GOSUB 5000:GOSUB 990:GOSUB 1110:GOS
UB 800:GOSUB 860:REM *** INITIALIZATIO
N ***
32 GOSUB 500:REM *** CLEAR REC$ ***
34 SAVC=PEEK(559):POKE 559,0:PRINT CHR
$(125):REM *** CLEAR SCREEN ***
36 GOSUB 210:GOSUB 360:POKE 559,SAVC:P
OKE 752,1:REM *** REC$ ON SCREEN ***
38 GOSUB 530:REM *** MENU AND CHOICE *
**
40 ON CHOICE GOSUB 300,500,60,70,82,40
0
42 GOTO 34
 0 ? CHR$(125):GOSUB 600:GOSUB 700:GOS
UB 740:IF C=0 THEN POSITION 5,10:? "NO
T FOUND":GOSUB 500:GOSUB 510:RETURN
62 S=715-C:GOSUB 850:RETURN
70 POSITION 3,18:? "WANT TO SAVE THI
S RECORD (Y/N)? ":GET #1,A:IF A()ASC("
Y") AND A()ASC("y") THEN RETURN
72 GOSUB 700:GOSUB 740:IF C()0 THEN GO
SUB 79:GET #1,A:IF A()ASC("Y") AND A()
ASC("y") THEN RETURN
73 IF C()0 THEN 76
74 GOSUB 720:S=FSEC:IF FSEC(33 THEN PO
SITION 3,18:? "                DISK FULL
          ":GOSUB 510:RETURN
76 GOSUB 800:GOSUB 750:INDEX$(IDX,IDX+
5)=CODE$:S=IDR:TEMP$=REC$
78 REC$=INDEX$(128*(IDR-1)+1,128*IDR):
GOSUB 750:REC$=TEMP$:RETURN
79 POSITION 3,18:? "      DUPLICATE -
REPLACE (Y/N)?    ":GOSUB 510:RETURN
82 POSITION 1,18:? "YOU WISH TO DELETE
 THIS RECORD (Y/N)?";:GET #1,A:IF A()A
SC("Y") AND A()ASC("y") THEN RETURN
84 GOSUB 700:GOSUB 740:IF C=0 THEN POS
ITION 1,18:? "                NOT FOUND
           ":GOSUB 510:RETURN
86 GOSUB 800:INDEX$(IDX,IDX+5)=FREX$:S
=IDR
88 REC$=INDEX$(128*(IDR-1)+1,128*IDR):
GOSUB 750:GOSUB 500:RETURN
99 REM * INPUT A STRING TEMP$ *
100 L=0:TEMP$="":POKE 764,255:USEFLG=0
110 IF FLIP=0 THEN FLIP=1:POKE 752,0:G
OTO 130
120 IF FLIP=1 THEN FLIP=0:POKE 752,1
130 IF PEEK(764)=255 THEN ? CHR$(31);C
HR$(30);:GOTO 110
140 GET #1,C:IF C=155 OR C=30 THEN RET
URN
150 USEFLG=1:IF C=126 THEN 180
160 IF L=LFLD THEN 110
164 IF L=0 THEN PRINT BLK$(1,LFLD):REC
$(POSFLD,POSFLD+LFLD-1)=BLK$:POSITION
XFLD,YFLD
170 L=L+1:? CHR$(C);:TEMP$(L,L)=CHR$(C
):GOTO 110
180 IF L)0 THEN L=L-1:? CHR$(C);:IF L=
0 THEN TEMP$=""
190 IF L)0 THEN TEMP$=TEMP$(1,L)
200 GOTO 110
209 REM * PUT FIELD INPUTS ON SCREEN *
210 RESTORE FLDLINE:READ NBRFLD:FOR J=
1 TO NBRFLD:READ POSFLD,LFLD,XFLD,YFLD
,NAFLD$
220 POSITION XFLD,YFLD+1:FOR K=1 TO LF
LD:? "-";:NEXT K:LNAM=LEN(NAFLD$)
230 POSITION XFLD+(LFLD-LNAM)/2,YFLD+1
:? NAFLD$;:NEXT J:RETURN
240 DATA 9
242 DATA 1,12,8,4,FIRST NAME
244 DATA 13,15,21,4,LAST NAME
246 DATA 28,24,8,6,SUB NAME
248 DATA 52,24,8,8,STREET ADDRESS
250 DATA 76,16,10,10,CITY
252 DATA 92,2,30,10,STATE
254 DATA 94,5,27,12,ZIPCODE
256 DATA 99,12,14,14,PHONE
258 DATA 111,17,8,16,EXTRA
299 REM * FILL REC$ FROM SCREEN INPUT
*
300 RESTORE FLDLINE:READ NBFLD:J=1
310 RESTORE FLDLINE+2*J:READ POSFLD,LF
LD,XFLD,YFLD,NAFLD$
320 POSITION XFLD,YFLD:GOSUB 100:IF US
EFLG=1 THEN GOSUB 350
322 IF C=30 AND J)1 THEN J=J-1:GOTO 31
0
330 J=J+1:IF J)NBRFLD THEN RETURN
340 GOTO 310
350 REC$(POSFLD,POSFLD+LEN(TEMP$))=TEM
P$:RETURN
359 REM *** PUT REC$ ON SCREEN ***
360 RESTORE FLDLINE:READ NBFLD:FOR J=1
 TO NBFLD:READ POSFLD,LFLD,XFLD,YFLD,N
AFLD$
370 POSITION XFLD,YFLD:? REC$(POSFLD,P
OSFLD+LFLD-1):NEXT J:RETURN
399 REM * PRINT LABEL ACCORDING TO DAT
A STATEMENTS *
400 ? CHR$(125):RESTORE PRTLINE:J=1:K=
-1:HOLD=0:SKIP=1
402 READ BLNKS,FLDNO:PRT$(J,J)=CHR$(BL
NKS):PRT$(J+1,J+1)=CHR$(FLDNO):IF FLDN
O=255 THEN 406
404 J=J+2:GOTO 402
406 ? #2;"      ";
410 K=K+2:BLNKS=ASC(PRT$(K)):FLDNO=ASC
(PRT$(K+1))
412 IF BLNKS=255 AND HOLD=0 THEN ? #2;
"":? #2;"      ";:GOTO 410
414 IF BLNKS=255 AND HOLD=1 THEN SKIP=
SKIP+1:GOTO 410
420 IF FLDNO=255 THEN ? #2;"":FOR J=1
TO SKIP:? #2;"":NEXT J:RETURN
428 IF BLNKS)0 THEN FOR J=1 TO BLNKS:?
 #2;" ";:NEXT J
430 RESTORE FLDLINE+2*FLDNO:READ POSFL
D,LFLD
432 L=POSFLD+LFLD-1
434 IF REC$(L,L)=" " AND L)POSFLD THEN
 L=L-1:GOTO 434
436 IF L=POSFLD THEN HOLD=1:GOTO 410
440 HOLD=0:? #2;REC$(POSFLD,L);:GOTO 4
10
460 DATA 0,1,1,2,255,0,0,3,255,0,0,4,2
55,0,0,5,3,6,255,0,14,7,0,255
500 REC$=" ":REC$(128)=REC$:REC$(2)=RE
C$:RETURN
510 SOUND 0,100,12,8:FOR K=1 TO 150:NE
XT K:SOUND 0,0,0,0:RETURN
529 REM *** MENU AND CHOICE INPUT ***
530 POSITION 2,20:? "┌──────────────────
──────────────┐"
532 POSITION 2,21:? "| (1)EDIT (2)CLEA
R (3)FIND (4)SAVE |"
534 POSITION 2,22:? "| (5)DELETE (6)PR
T LABEL  Choice?  |"
536 POSITION 2,23:? "└──────────────────
──────────────┘";
540 GET #1,CHOICE:CHOICE=CHOICE-48:IF
CHOICE(1 OR CHOICE)6 THEN 540
550 RETURN
599 REM *** GET NAME ONLY FROM SCREEN
***
600 RESTORE FLDLINE:READ NBRFLD:FOR J=
1 TO 2:READ POSFLD,LFLD,XFLD,YFLD,NAFL
D$
610 POSITION XFLD,YFLD+1:FOR K=1 TO LF
LD:? "-";:NEXT K:LNAM=LEN(NAFLD$)
```

by STAN OCKERS

```
830 ASECRW=ADR(SECRW$):XADR=ADR(X55$):
FOR J=1 TO 32
840 IO=USR(ASECRW,XADR,J,1):NEXT J:IO=
USR(ASECRW,XADR,720,1):RETURN
849 REM *** READ SECTOR 'S' INTO REC$
***
850 REC$(128)=" ":AREC=ADR(REC$):ASECR
W=ADR(SECRW$):IO=USR(ASECRW,AREC,S,0):
RETURN
859 REM *** READ SECTORS 1-32 INTO IND
EX ***
860 INDEX$=" ":INDEX$(4096)=INDEX$:IND
EX$(2)=INDEX$
870 FOR S=1 TO 32:GOSUB 850:J=(S-1)*12
8+1:INDEX$(J,J+127)=REC$:NEXT S:RETURN

879 REM *** SAVE INDEX$ TO SECTORS 1-3
2 ***
880 FOR S=1 TO 32:J=(S-1)*128+1:REC$=I
NDEX$(J,J+127):GOSUB 750:NEXT S:RETURN

980 REM *** TOTAL STRING SEARCH ***
982 REM *** ANALOG #12 P. 84    ***
984 REM *** C=USR(ADR(B$),CNT,ADR(A$),
ADR(DT$)RL,DTL) ***
985 REM *** CNT=RECORD CNT ***
986 REM *** A$=STRING DT$=DESIRED TERM
 ***
987 REM *** RL=RECORD LEN DTL=DES TERM
 LEN ***
990 DIM B$(139):RESTORE 1000:FOR J=1 T
O 139:READ A:B$(J,J)=CHR$(A):NEXT J:RE
TURN
1000 DATA 216,104,104,133,204,104,133,
203,104,133,209,104,133,208,104,133,21
5,104,133,214
1010 DATA 104,104,133,205,104,104,133,
206,169,0,133,212,169,0,133,213,162,0,
160,0
1020 DATA 177,214,224,0,208,2,132,216,
209,208,208,43,232,228,206,240,22,200,
196,205
1030 DATA 240,50,72,152,72,138,168,177
,214,133,207,104,168,104,165,207,24,14
4,219,72
1040 DATA 165,204,133,213,165,203,133,
212,104,162,0,224,0,240,17,224,0,240,6
,160
1050 DATA 0,177,214,162,0,164,216,200,
196,205,208,186,165,208,24,101,205,133
,208,144
1060 DATA 2,230,209,165,203,208,6,165,
204,240,7,198,204,198,203,24,144,156,9
6,-1
620 POSITION XFLD+(LFLD-LNAM)/2,YFLD+1
:? NAFLD$;:NEXT J
630 RESTORE FLDLINE:GOSUB 500:READ NBF
LD:J=1
640 RESTORE FLDLINE+2*J:READ POSFLD,LF
LD,XFLD,YFLD,NAFLD$
650 POSITION XFLD,YFLD:GOSUB 100:IF US
EFLG=1 THEN GOSUB 350
660 IF C=30 AND J>1 THEN J=J-1:GOTO 64
0
670 J=J+1:IF J>2 THEN RETURN
680 GOTO 640
699 REM *** MAKE A CODE STRING FROM NA
MES ***
700 CODE$="":RESTORE FLDLINE+4:READ J:
RESTORE FLDLINE+2:READ K:CODE$=REC$(J,
J+2):L=4
702 IF CODE$(3,3)=" " THEN CODE$=CODE$
(1,2):L=3
704 IF CODE$(2,2)=" " THEN CODE$=CODE$
(1,1):L=2
706 CODE$(L,L+1)=REC$(K,K+1):IF CODE$(
L+1,L+1)<>" " THEN RETURN
707 IF CODE$(L+1,L+1)=" " THEN CODE$=C
ODE$(1,L)
708 IF CODE$(L,L)=" " THEN CODE$=CODE$
(1,L-1)
710 RETURN
719 REM *** FIND 1ST FREE POSITION IN
INDEX$ ***
720 C=USR(ADR(B$),681,ADR(INDEX$),ADR(
FREX$),6,5):FSEC=714-C:IDX=4087-C*6:ID
R=INT(IDX/128)+1:RETURN
739 REM *** FIND CODE STRING IN INDEX$
 RET. SECTOR 'S' ***
740 C=USR(ADR(B$),682,ADR(INDEX$),ADR(
CODE$),6,LEN(CODE$)):S=715-C:IDX=4093-
C*6:IDR=INT(IDX/128)+1:RETURN
749 REM *** SAVE REC$ TO SECTOR S ***
750 IO=USR(ADR(SECRW$),ADR(REC$),S,1):
RETURN
799 REM *** WARNING  ***
800 TEMP$=REC$:SAVS=S:S=720:GOSUB 850:
IF REC$(1,10)=X55$(1,10) THEN REC$=TEM
P$:S=SAVS:RETURN
802 REC$=TEMP$:S=SAVS:POSITION 3,10:?
"WARNING!!- BE SURE DATA DISK IS IN":P
OSITION 10,12
810 ? "THEN PRESS ANY KEY":POKE 764,25
5
820 IF PEEK(764)=255 THEN 820
822 RETURN
829 REM *** REM FILL DISK SECTORS 1-32
 & 720 WITH X'S ***
1100 REM * LIVE W/O DOS ANALOG #17 P 5
4 *
1102 REM * IO=USR(ADR(SECRW$),ADR(BUF$
),SECT,FLAG *
1104 REM * FLAG=8 TO READ, 1 TO WRITE
*
1110 DIM SECRW$(44):RESTORE 1120:FOR J
=1 TO 44:READ A:SECRW$(J,J)=CHR$(A):NE
XT J:RETURN
1120 DATA 104,104,141,5,3,104,141,4,3,
104,141,11,3,104,141,10,3,104,104,201
1130 DATA 1,208,7,169,87,141,2,3,208,5
,169,82,141,2,3,169,1,141,1,3,32,83,22
8,96
5000 DIM TEMP$(128),NAFLD$(20),REC$(12
8),BLK$(50),X55$(128):LFLD=18:? CHR$(1
25)
5010 X55$="X":X55$(128)=X55$:X55$(2)=X
55$:FLDLINE=240:PRTLINE=460:DIM PRT$(5
0),INDEX$(4096)
5020 DIM FREX$(6):FREX$="XXXXXX":DIM C
ODE$(6):OPEN #1,4,0,"K:":OPEN #2,8,0,"
P:"
5030 BLK$=" ":BLK$(50)=BLK$:BLK$(2)=BL
K$:A=PEEK(16):IF A>128 THEN A=A-128:PO
KE 16,A:POKE 53774,A:RETURN
```

MARCH

MEETING

WED. the 13th

SOUTH EUG.

7:30pm

SNAKES by Lex Johnson

```
0 GRAPHICS 7+16:COLOR 3
1 DIM MAN(8,8):DIM ATARI(8,8):DIM MANS
POT(8,8):DIM ATSPOT(8,8)
2 FOR I=0 TO 7:FOR J=0 TO 7
3 MAN(I,J)=0:ATARI(I,J)=0:MANSPOT(I,J
)=0:ATSPOT(I,J)=0
4 NEXT J:NEXT I
5 CHECK=0
50 GRAPHICS 2+16
70 POSITION 0,3:? #6;"       sNAKES"
80 POSITION 9,5:? #6;"AND"
90 POSITION 7,7:? #6;"1ADDERS"
100 POSITION 3,1:PRINT #6;"0_"
110 FOR I=2 TO 10 STEP 2
120 POSITION 2,I:? #6;"(":NEXT I
130 FOR I=3 TO 9 STEP 2
140 POSITION 3,I:? #6;")":NEXT I
150 FOR I=1 TO 10
160 POSITION 17,I:? #6;"H":NEXT I
170 FOR ZZ=1 TO 900:NEXT ZZ
180 POSITION 3,11:? #6;"BY LEX JOHNSON
"
260 FOR ZZ=1 TO 1000:NEXT ZZ
270 REM INSTRUCTIONS FOR USE
280 REM DO NOT FEED AFTER MIDNIGHT !!!
290 GRAPHICS 2+16
300 POSITION 3,0:PRINT #6;"instruction
s"
310 ? #6;"THE IDEA OF THE GAME IS TO G
ET YOUR MAN  TO THE LAST SQUARE  QUICK
LY"
320 ? #6;"YOU CAN CLIMB UP THE RED LAD
DERS BUT     LOOK OUT FOR THE    GREEN
 SNAKES' HEADS ...YOU'LL FALL!"
330 ? #6;"PRESS SELECT"
340 POKE 53279,0
350 IF PEEK(53279)=5 THEN 370
360 GOTO 350
370 GRAPHICS 2+16
380 ? #6;" more instructions"
390 ? #6;"YOU MUST TAKE TURNS  WITH TH
E COMPUTER."
400 ? #6;"SELECT THE NUMBER    THROWN
BY YOUR DICE WHEN THE BUZZER     SOUND
S AND THE      RIGHT HAND SIDE OF"
410 ? #6;" THE BOARD IS BLANK.THE COMP
UTER HAS THE FIRST GO.          PRESS
SELECT"
420 POKE 53279,0
430 IF PEEK(53279)=5 THEN 450
440 GOTO 430
450 GRAPHICS 2+16
460 POSITION 0,5:? #6;"       PRESS
          START"
470 POKE 53279,0
480 IF PEEK(53279)=6 THEN 500
490 GOTO 480
500 GRAPHICS 7+16:FOR X=35 TO 125 STEP
 9
505 COLOR 3
510 PLOT X,0:DRAWTO X,90
520 NEXT X
530 FOR Y=0 TO 90 STEP 9
540 PLOT 35,Y:DRAWTO 125,Y
550 NEXT Y
600 REM SPACE FOR DRAWING NUMBERS
610 COLOR 3
620 REM DRAW ALL 1'S
625 RESTORE 670
630 FOR I=1 TO 13
640 READ X,Y:PLOT X,Y:DRAWTO X,Y+4
650 NEXT I
660 REM DATA FOR 1'S
670 DATA 37,3,123,3,42,12,123,21,42,30
,123,39,42,48,123,57,42,66,118,84,40,8
4,121,75,118,75
680 FOR I=1 TO 8
690 PLOT I*9+37,75:DRAWTO I*9+37,79
700 NEXT I
710 REM DRAW ALL THE 2'S
715 RESTORE 760
720 FOR I=1 TO 11
730 READ X,Y
740 PLOT X,Y:DRAWTO X+2,Y:DRAWTO X+2,Y
+2:DRAWTO X,Y+2:DRAWTO X,Y+4:DRAWTO X+
2,Y+4
750 NEXT I
760 DATA 113,3,113,21,113,39,113,57,11
2,75,50,12,50,30,50,48,50,66,48,84,37,
75
770 FOR I=0 TO 8
780 X=I*9+37:Y=66
790 PLOT X,Y:DRAWTO X+2,Y:DRAWTO X+2,Y
+2:DRAWTO X,Y+2:DRAWTO X,Y+4:DRAWTO X+
2,Y+4
800 NEXT I
810 REM DRAW ALL THE 3'S
812 RESTORE 835
815 FOR I=1 TO 11
820 READ X,Y
825 PLOT X,Y:DRAWTO X+2,Y:DRAWTO X+2,Y
+4:DRAWTO X,Y+4:PLOT X+2,Y+2:DRAWTO X,
Y+2
830 NEXT I
835 DATA 104,3,104,21,104,39,104,57,10
3,75,59,12,59,30,59,48,59,66,57,84,118
,66
840 FOR I=0 TO 8
845 X=(118-I*9):Y=57
850 PLOT X,Y:DRAWTO X+2,Y:DRAWTO X+2,Y
+4:DRAWTO X,Y+4:PLOT X+2,Y+2:DRAWTO X,
Y+2
855 NEXT I
860 REM DRAW ALL THE 4'S
862 RESTORE 885
865 FOR I=1 TO 11
870 READ X,Y
875 PLOT X,Y:DRAWTO X,Y+3:DRAWTO X+2,Y
+3:PLOT X+2,Y+2:DRAWTO X+2,Y+4
880 NEXT I
885 DATA 95,3,95,21,95,39,95,57,94,75,
68,12,68,30,68,48,68,66,66,84,37,57
890 FOR I=0 TO 8
895 X=I*9+37:Y=48
900 PLOT X,Y:DRAWTO X,Y+3:DRAWTO X+2,Y
+3:PLOT X+2,Y+2:DRAWTO X+2,Y+4
905 NEXT I
910 REM DRAW ALL THE 5'S
912 RESTORE 935
915 FOR I=1 TO 11
920 READ X,Y
925 PLOT X+2,Y:DRAWTO X,Y:DRAWTO X,Y+2
:DRAWTO X+2,Y+2:DRAWTO X+2,Y+4:DRAWTO
X,Y+4
930 NEXT I
935 DATA 86,3,86,21,86,39,86,57,85,75,
77,12,77,30,77,48,77,66,75,84,118,48
940 FOR I=0 TO 8
945 X=(118-I*9):Y=39
950 PLOT X+2,Y:DRAWTO X,Y:DRAWTO X,Y+2
:DRAWTO X+2,Y+2:DRAWTO X+2,Y+4:DRAWTO
X,Y+4
955 NEXT I
960 REM DRAW ALL THE 6'S
962 RESTORE 985
965 FOR I=1 TO 11
970 READ X,Y
975 PLOT X+2,Y:DRAWTO X,Y:DRAWTO X,Y+4
:DRAWTO X+2,Y+4:DRAWTO X+2,Y+2:DRAWTO
X,Y+2
980 NEXT I
985 DATA 77,3,77,21,77,39,77,57,76,75,
86,12,86,30,86,48,86,66,84,84,37,39
990 FOR I=0 TO 8
995 X=I*9+37:Y=30
1000 PLOT X+2,Y:DRAWTO X,Y:DRAWTO X,Y+
4:DRAWTO X+2,Y+4:DRAWTO X+2,Y+2:DRAWTO
 X,Y+2
1005 NEXT I
1010 REM DRAW ALL THE 7'S
1012 RESTORE 1035
1015 FOR I=1 TO 11
```

snakes cont

```
1020 READ X,Y
1025 PLOT X,Y:DRAWTO X+2,Y:DRAWTO X+2,
Y+2:PLOT X+1,Y+2:DRAWTO X+1,Y+4
1030 NEXT I
1035 DATA 68,3,68,21,68,39,68,57,67,75
,95,12,95,30,95,48,95,66,93,84,118,30
1040 FOR I=0 TO 8
1045 X=(118-I*9):Y=21
1050 PLOT X,Y:DRAWTO X+2,Y:DRAWTO X+2,
Y+2:PLOT X+1,Y+2:DRAWTO X+1,Y+4
1055 NEXT I
1060 REM DRAW ALL THE 8'S
1062 RESTORE 1085
1065 FOR I=1 TO 11
1070 READ X,Y
1075 PLOT X,Y:DRAWTO X+2,Y:DRAWTO X+2,
Y+4:DRAWTO X,Y+4:DRAWTO X,Y:PLOT X,Y+2
:DRAWTO X+2,Y+2
1080 NEXT I
1085 DATA 59,3,59,21,59,39,59,57,58,75
,104,12,104,30,104,48,104,66,102,84,37
,21
1090 FOR I=0 TO 8
1095 X=I*9+37:Y=12
1100 PLOT X,Y:DRAWTO X+2,Y:DRAWTO X+2,
Y+4:DRAWTO X,Y+4:DRAWTO X,Y:PLOT X,Y+2
:DRAWTO X+2,Y+2
1105 NEXT I
1110 REM DRAW ALL THE 9'S
1112 RESTORE 1135
1115 FOR I=1 TO 11
1120 READ X,Y
1125 PLOT X+2,Y+2:DRAWTO X,Y+2:DRAWTO
X,Y:DRAWTO X+2,Y:DRAWTO X+2,Y+4:DRAWTO
 X,Y+4
1130 NEXT I
1135 DATA 50,3,50,21,50,39,50,57,49,75
,113,12,113,30,113,48,113,66,111,84,11
8,12
1140 FOR I=0 TO 8
1145 X=(118-I*9):Y=3
1150 PLOT X+2,Y+2:DRAWTO X,Y+2:DRAWTO
X,Y:DRAWTO X+2,Y:DRAWTO X+2,Y+4:DRAWTO
 X,Y+4
1155 NEXT I
1160 REM DRAW ALL THE 0'S
1162 RESTORE 1185
1165 FOR I=1 TO 11
1170 READ X,Y
1175 PLOT X,Y:DRAWTO X,Y+4:DRAWTO X+2,
Y+4:DRAWTO X+2,Y:DRAWTO X,Y
1180 NEXT I
1185 DATA 39,3,41,3,41,21,41,39,41,57,
40,75,122,12,122,30,122,48,122,66,120,
84
1400 REM DRAW THE SNAKES
1410 COLOR 2
1420 FOR J=1 TO 12
1430 READ X1,Y1,X2,Y2
1440 PLOT X1,Y1
1445 PLOT X2-1,Y2:DRAWTO X2+1,Y2
1446 PLOT X2-1,Y2+1:DRAWTO X2+1,Y2+1
1450 FOR I=1 TO Y1-Y2
1460 Y=Y1-I
1470 X=INT((X2-X1)*(Y1-Y)/(Y1-Y2)+X1)
1480 IF RND(0)<0.5 THEN X=X-1
1490 PLOT X,Y
1500 NEXT I
1510 NEXT J
1520 DATA 42,83,100,34
1530 DATA 60,83,82,70
1540 DATA 69,83,73,52
1550 DATA 109,83,91,52
1560 DATA 118,83,87,34
1570 DATA 105,74,118,25
1580 DATA 55,65,42,34
1590 DATA 64,56,51,7
1600 DATA 118,47,109,7
1610 DATA 109,38,69,7
1620 DATA 69,29,82,7
1630 DATA 42,74,64,16
1700 REM TO DRAW THE LADDERS
1710 COLOR 1
1720 PLOT 55,74:DRAWTO 51,61:DRAWTO 46
,61
1730 PLOT 87,83:DRAWTO 91,61:DRAWTO 96
,61
1740 PLOT 118,65:DRAWTO 105,43:DRAWTO
100,43
1750 PLOT 42,56:DRAWTO 55,16:DRAWTO 60
,16
1760 PLOT 60,47:DRAWTO 64,25:DRAWTO 69
,25
1770 PLOT 118,38:DRAWTO 105,16:DRAWTO
100,16
1780 PLOT 87,38:DRAWTO 100,7:DRAWTO 10
5,7
1790 PLOT 73,29:DRAWTO 78,16:DRAWTO 73
,16
1800 PLOT 46,20:DRAWTO 42,7:DRAWTO 37,
7
2000 FOR I=0 TO 7:FOR J=0 TO 7
2010 READ A:MAN(I,J)=A
2020 NEXT J:NEXT I
2030 DATA 0,0,0,1,1,0,0,0
2040 DATA 0,0,0,1,1,0,0,0
2050 DATA 0,0,0,1,1,0,0,0
2060 DATA 1,1,1,1,1,1,1,1
2070 DATA 1,1,1,1,1,1,1,1
2080 DATA 0,0,0,1,1,0,0,0
2090 DATA 0,0,0,1,1,0,0,0
2100 DATA 0,0,0,1,1,0,0,0
2200 FOR I=0 TO 7:FOR J=0 TO 7
2210 READ A:ATARI(I,J)=A
2220 NEXT J:NEXT I
2230 DATA 1,1,1,0,0,1,1,1
2240 DATA 1,1,1,0,0,1,1,1
2250 DATA 1,1,1,0,0,1,1,1
2260 DATA 0,0,0,0,0,0,0,0
2270 DATA 0,0,0,0,0,0,0,0
2280 DATA 1,1,1,0,0,1,1,1
2290 DATA 1,1,1,0,0,1,1,1
2300 DATA 1,1,1,0,0,1,1,1
2400 MANX=27:MANY=82:ATARIX=27:ATARIY=
82
2410 MAN=0:ATARI=0
2500 PLUS=INT(6*RND(0)+1)
2510 COLOR 3
2520 FOR I=1 TO PLUS
2530 PLOT 25,I*10
2540 NEXT I
2545 IF ATARI+PLUS>100 THEN 4000
2550 FOR I=0 TO 7
2560 FOR J=0 TO 7
2570 COLOR ATSPOT(I,J)
2580 PLOT ATARIX+I,ATARIY+J
2590 NEXT J
2600 NEXT I
2601 IF DRAWMAN<>1 THEN 2609
2602 COLOR 1
2603 FOR I=0 TO 7
2604 FOR J=0 TO 7
2605 IF MAN(I,J)=1 THEN PLOT ATARIX+I,
ATARIY+J
2606 NEXT J
2607 NEXT I
2608 DRAWMAN=0
2609 IF CHECK=1 THEN 2620
2610 ATARI=ATARI+PLUS
2620 IF ATARI<>MAN THEN 2670
2625 DRAWAT=1
2630 FOR I=0 TO 7:FOR J=0 TO 7
2640 ATSPOT(I,J)=MANSPOT(I,J)
2650 NEXT J:NEXT I:ATARIX=MANX:ATARIY=
MANY
2660 GOTO 3000
2670 ATARIY=(82-INT((ATARI-1)/10)*9)
2680 IF (INT((ATARI-1)/10)/2)=INT(INT(
(ATARI-1)/10)/2) THEN ATARIX=(((((ATAR
I-1)/10)-INT((ATARI-1)/10))*10)+1)*9+2
7
2690 IF (INT((ATARI-1)/10)/2)<>INT(INT
((ATARI-1)/10)/2) THEN ATARIX=126-((((
(ATARI-1)/10)-INT((ATARI-1)/10))*10)+1
)*9
2700 FOR I=0 TO 7
2710 FOR J=0 TO 7
2720 LOCATE ATARIX+I,ATARIY+J,Q
2730 ATSPOT(I,J)=Q
2740 NEXT J
2750 NEXT I
```

snakes cont

```
3000 COLOR 1
3010 FOR I=0 TO 7
3020 FOR J=0 TO 7
3030 IF ATARI(I,J)=1 THEN PLOT ATARIX+
I,ATARIY+J
3040 NEXT J
3050 NEXT I
3100 COLOR 0:PLOT 25,0:DRAWTO 25,95
3110 IF CHECK=1 THEN CHECK=0:GOTO 3400
3200 IF ATARI=99 THEN ATARI=37:CHECK=1
:GOTO 3600
3210 IF ATARI=97 THEN ATARI=52:CHECK=1
:GOTO 3600
3220 IF ATARI=95 THEN ATARI=64:CHECK=1
:GOTO 3600
3230 IF ATARI=92 THEN ATARI=50:CHECK=1
:GOTO 3600
3240 IF ATARI=84 THEN ATARI=20:CHECK=1
:GOTO 3600
3250 IF ATARI=71 THEN ATARI=13:CHECK=1
:GOTO 3600
3260 IF ATARI=68 THEN ATARI=1:CHECK=1:
GOTO 3600
3270 IF ATARI=66 THEN ATARI=10:CHECK=1
:GOTO 3600
3280 IF ATARI=61 THEN ATARI=23:CHECK=1
:GOTO 3600
3290 IF ATARI=47 THEN ATARI=9:CHECK=1:
GOTO 3600
3300 IF ATARI=45 THEN ATARI=4:CHECK=1:
GOTO 3600
3310 IF ATARI=26 THEN ATARI=3:CHECK=1:
GOTO 3600
3400 IF CHECK=1 THEN CHECK=0
3410 IF ATARI=6 THEN ATARI=34:CHECK=1:
GOTO 3700
3420 IF ATARI=18 THEN ATARI=39:CHECK=1
:GOTO 3700
3430 IF ATARI=30 THEN ATARI=53:CHECK=1
:GOTO 3700
3440 IF ATARI=40 THEN ATARI=83:CHECK=1
:GOTO 3700
3450 IF ATARI=43 THEN ATARI=77:CHECK=1
:GOTO 3700
3460 IF ATARI=51 THEN ATARI=88:CHECK=1
:GOTO 3700
3470 IF ATARI=55 THEN ATARI=93:CHECK=1
:GOTO 3700
3480 IF ATARI=65 THEN ATARI=85:CHECK=1
:GOTO 3700
3490 IF ATARI=79 THEN ATARI=100:CHECK=
1:GOTO 3700
3500 IF ATARI=100 THEN 7000
3550 GOTO 4000
3600 REM SNAKE SOUND
3610 RESTORE 3680
3620 FOR I=1 TO 16:READ TONE
3630 SOUND 0,TONE,10,8:FOR ZZ=1 TO 1:N
EXT ZZ
3640 NEXT I
3650 SOUND 0,0,0,0:FOR ZZ=1 TO 10:NEXT
 ZZ
3660 SOUND 0,243,12,15:FOR ZZ=1 TO 5:N
EXT ZZ
3670 SOUND 0,0,0,0
3680 DATA 40,45,50,57,64,72,81,91,102,
114,128,144,162,182,204,230
3690 GOTO 2550
3700 REM LADDER SOUND
3710 FOR I=1 TO 10
3720 SOUND 0,29,8,15
3730 FOR ZZ=1 TO 1:NEXT ZZ
3740 SOUND 0,0,0,0
3750 FOR ZZ=1 TO 3:NEXT ZZ
3760 NEXT I
3770 GOTO 2550
4000 COLOR 0:PLOT 135,0:DRAWTO 135,90
4010 COLOR 3
4020 POKE 53279,0
4030 PLUS=INT(6*RND(0)+1)
4040 IF PEEK(53279)=5 THEN 4060
4050 GOTO 4020
4060 FOR I=1 TO PLUS:PLOT 135,I*10:NEX
T I
4080 IF MAN+PLUS)100 THEN 2500
4100 PLUS=PLUS
4400 FOR I=0 TO 7
4410 FOR J=0 TO 7
4420 COLOR MANSPOT(I,J)
4430 PLOT MANX+I,MANY+J
4440 NEXT J
4450 NEXT I
4460 IF DRAWAT()1 THEN 4535
4470 COLOR 1
4480 FOR I=0 TO 7
4490 FOR J=0 TO 7
4500 IF ATARI(I,J)=1 THEN PLOT MANX+I,
MANY+J
4510 NEXT J
4520 NEXT I
4530 DRAWAT=0
4535 IF CHECK=1 THEN 4550
4540 MAN=MAN+PLUS
4550 IF MAN()ATARI THEN 4600
4555 DRAWMAN=1
4560 FOR I=0 TO 7:FOR J=0 TO 7
4570 MANSPOT(I,J)=ATSPOT(I,J)
4580 NEXT J:NEXT I:MANX=ATARIX:MANY=AT
ARIY
4590 GOTO 4790
4600 MANY=(82-INT((MAN-1)/10)*9)
4610 IF (INT((MAN-1)/10)/2)=INT(INT((M
AN-1)/10)/2) THEN MANX=(((((MAN-1)/10)
-INT((MAN-1)/10))*10)+1)*9+27
4620 IF (INT((MAN-1)/10)/2)()INT(INT((
MAN-1)/10)/2) THEN MANX=126-(((((MAN-1
)/10)-INT((MAN-1)/10))*10)+1)*9
4700 FOR I=0 TO 7
4710 FOR J=0 TO 7
4720 LOCATE MANX+I,MANY+J,Q
4730 MANSPOT(I,J)=Q
4740 NEXT J
4750 NEXT I
4790 COLOR 1
4800 FOR I=0 TO 7
4810 FOR J=0 TO 7
4820 IF MAN(I,J)=1 THEN PLOT MANX+I,MA
NY+J
4830 NEXT J
4840 NEXT I
5100 COLOR 0:PLOT 25,0:DRAWTO 25,95
5110 IF CHECK=1 THEN CHECK=0:GOTO 5400
5200 IF MAN=99 THEN MAN=37:CHECK=1:GOT
O 5600
5210 IF MAN=97 THEN MAN=52:CHECK=1:GOT
O 5600
5220 IF MAN=95 THEN MAN=64:CHECK=1:GOT
O 5600
5230 IF MAN=92 THEN MAN=50:CHECK=1:GOT
O 5600
5240 IF MAN=84 THEN MAN=20:CHECK=1:GOT
O 5600
5250 IF MAN=71 THEN MAN=13:CHECK=1:GOT
O 5600
5260 IF MAN=68 THEN MAN=1:CHECK=1:GOTO
 5600
5270 IF MAN=66 THEN MAN=10:CHECK=1:GOT
O 5600
5280 IF MAN=61 THEN MAN=23:CHECK=1:GOT
O 5600
5290 IF MAN=47 THEN MAN=9:CHECK=1:GOTO
 5600
5300 IF MAN=45 THEN MAN=4:CHECK=1:GOTO
 5600
5310 IF MAN=26 THEN MAN=3:CHECK=1:GOTO
 5600
5400 IF CHECK=1 THEN CHECK=0
5410 IF MAN=6 THEN MAN=34:CHECK=1:GOTO
 5700
5420 IF MAN=18 THEN MAN=39:CHECK=1:GOT
O 5700
5430 IF MAN=30 THEN MAN=53:CHECK=1:GOT
O 5700
5440 IF MAN=40 THEN MAN=83:CHECK=1:GOT
O 5700
5450 IF MAN=43 THEN MAN=77:CHECK=1:GOT
O 5700
5460 IF MAN=51 THEN MAN=88:CHECK=1:GOT
O 5700
5470 IF MAN=55 THEN MAN=93:CHECK=1:GOT
O 5700
```

LABELS

Recently, I find myself mailing a fairly large number of checks each month. While addressing envelopes, I suddenly thought "wouldn't it be nice if I had a program where I could type in a few letters and have the program dig out the address and print a mailing label".

The result is a program I call 'Labels'. Labels uses a data disk holding up to 681 addresses, each on one sector of the data disk (128 bytes). 32 sectors of the disk hold an index which is loaded in memory at the beginning of a session and allows rapid location of any address.

The key to finding an address is based on the first three letters of the last name plus the first two of the first name. An index is held in a string called INDEX$, each address being allotted six bytes, (five for the key plus one for expansion). Two programs from ANALOG magazine were vital to creating 'Labels', (they are referenced in the listing). One allows writing to disk without DOS and the other permits rapid string searches.

When the program is run, you will be presented with a screen of labeled fields and a menu of choices. Choice #1 (EDIT) will produce a cursor in the first-name field. When you enter the first name and press return the cursor will jump to the second field (last-name). If you want to skip over a field, just press return. You can use CTRL backarrow to back up for re-editing. When the last field is entered, you will be returned to the menu.

Other menu choices are: (2) CLEAR to clear all fields to blanks; (3) FIND to search for an address; (4) SAVE to save what is currently displayed on the screen; (5) DELETE to delete the address currently displayed; and (6) PRT LABEL to print a label of what is displayed.

When you save an address a check is first made to determine if one with the same key exists. You will then be asked if you want to replace the current address (you can't have two with the same key). If you have updated an old address, the answer is yes, but if you haven't entered this name before, obviously some other name reduces to the same key. This might happen for example with John Jones Jr. and John Jones Sr. You will just have to find a way to change the name, putting a blank at the beginning of the first or last name for example. You'll have to remember to enter the blanks when searching for the name.

When you use option 3 (FIND) you will need to enter parts of last/first names. Just remember how the key is made. The last name is the primary identification. If three letters are available, they will all be used but as few as one could be used. In all cases the last occurance of a match in INDEX$ will be found. If a first name is supplied, as many as two of the first letters will be used. Leading blanks are significant but trailing ones are ignored. Upper and lowercase are also significant. INDEX$ can be viewed by hitting system reset and printing it. Single names like 'Wiebolts' should appear in the last name field.

This program uses a specially formatted data disk. First format the disk normally. Then run the 'Labels' program and press system reset. Make sure the data disk is in the drive and enter GOSUB 830. The disk will have X's written into sectors 1-32 (the index) and sector 720. Use only this disk with the program 'Labels'. The program writes directly to disk sectors and can easily mess up a normal disk.

If you don't like the field arrangement, it can easily be changed in lines 240 and following. The fields are specified in order in even numbered lines 242 and following (240 contains the number of fields). Each data statement contains the position in the record, the length of the field, the X screen position, Y screen position and the name of the field. You can have as many fields as you want but the total of all lengths can't exceed 128. Be sure you finalize the new arrangement before you start entering addresses. If you change later, they won't be compatible.

Printing of labels is also controlled by a data statement (line 460). The arrangement is a sequence of two number values: number of blanks followed by a field number. If a 255 appears in the # blanks position it indicates a carriage return is to take place. A 255 in the field number place indicates the printing is finished.

Only simple label printing is provided. The program assumes single width labels of size 3-1/2 x 15/16 inches. These will hold 5 lines and are separated by one line spacing. After the five, the program will bring the printer to the next label. You will have to manually position at the top of the first label where the 1st line goes. The lines are left justified. The program keeps track if fewer than five lines are printed and adjusts accordingly.

Some enhancements to the program might be considered. I left the 6th byte for each sector entry in the index with the idea it could be used to tag labels to be printed in separate lists of addresses. Using a separate bit for each list, you could have 8 lists. When the name of a list is entered the program will go through and print a label for each address for which the appropriate bit is set. Another mod might be to have the program print a directory of addresses, perhaps with the list finding provision and maybe in alphabetical order. A reasonably fast sorting routine working on the index would be required.

— Stan Ockers

STRING MAGIC

(reprint: STARFLEET, December, 1984)

Page Flipping Magic

Page flipping is the rapid changing of what is displayed on the screen between two or more images. This gives the illusion of movement, much like motion pictures. While there are several methods for accomplishing this, the example program will show you how you can use the string magic method.

While the program is longer than the other two examples, the concept is really quite simple. First, draw an image on the screen. Save the screen image in a string. Draw the next image on the screen. Save that in a string, and so forth until all the different images have been saved in strings. Then display the strings on the screen one after the other to give the illusion of movement.

Here is how you do it. Line 50 dimensions a string named DM$. We'll use the string magic routine to point this string to the display memory area. The strings SM1$, SM2$ and SM3$ will be used to save the different versions of display memory. Each string is dimensioned for only 480 bytes instead of 3840 (the size of a mode 23 screen) because we will save only the first 480 bytes of the screen.

Lines 90 through 110 invoke graphics mode 23 (mode 7 without a text window), set the background to dark blue and set our image (this time a bat) to black. The POKE 559,0 statement on Line 120 turns off the display. This is done for two reasons. First so you can't see the different versions of the bat being drawn on the screen. Second, to make the program run faster while the different versions are being drawn. If you want to see the different versions drawn on the screen, omit Line 120 or change it to a REMARK.

The parameters for the string magic routine are set up on Lines 150 through 170. The ADDR parameter is set to point to display memory and the SIZE parameter is set to 3840 — the size of a mode 23 screen. After the string magic routine is called by the GOSUB on Line 190, the first version of the bat is drawn four times across the top of the screen. This is done on Lines 220 through 260. Next, Line 280 saves the first 480 bytes of display memory into the string SM1$. Since the variable DM$ now points to the display memory area, the statement SM1$ = DM$ will move the first 480 bytes (the dimensioned size of SM1$) of display memory into SM1$.

On Line 300 is the same trick used in example program number 2. Here the statement DM$(2) = DM$(1) will move whatever is in position one of the string (in this case background color) to the rest of the string. This results in the entire screen being cleared to the background color.

The process of drawing four bats across the screen, saving the display in a string and clearing the display, is repeated until all three versions of the bat are saved in the strings SM1$, SM2$ and SM3$. This is done on Lines 220 through 480.

Since the head and eyes of the bat are the same in each version, the subroutine in Lines 660 through 730 is used to draw the head and eyes. The POKE 559,34 statement on Line 500 turns back on the display.

Now we're ready to perform the animation by flipping through the different versions. Line 520 starts the process by moving the first version stored in string SM1$ into the first 480 bytes of display memory. The second version of the bat stored in SM2$ is moved into the next 480 bytes of display memory. The subroutine in Line 750 moves the first 960 bytes of display memory into the next 960 bytes of display memory, then the first 1920 bytes to the next 1920 bytes. This gives us a full screen of bats. The subroutine also changes the color of the bats' eyes just for effect. The process of moving different versions of the bat to display memeory is repeated indefinitely by Lines 520 through 640, giving the illusion of continuous movement.

The Magic Revealed

If you don't really care exactly how the string magic routine works then yu won't need to read the rest of this section. Just remember the string magic routine can be used anytime you need to quickly move or modify sections of memory. There are many more applications for the routine — just use your imagination.

Before you can understand exactly how the string magic routine works, you must first understand how BASIC stores information about variables used in a program. For each variable you enter BASIC creates an entry in a table called the Variable Value Table (VVT). The address of the VVT is always stored in locations 134 and 135 with the low byte first. You can get this address with the following statement: VVT = PEEK(134) + PEEK(135)*256.

Each entry in the table is eight bytes long. The information in these eight bytes varies depending on the type of variable (simple, array or string). If the variable is simple the value of the variable is stored right in the table entry. However, if it's a string or an array variable, then the entry contains information about where in the string/array area the string or array can be found. See Table 1 for the format of the VVT entry for strings.

Armed with this knowledge, we are now ready to demystify the string magic routine. This is the first statement: A = (ADDR-(PEEK(140) + PEEK(141)*256)). This gets the difference between the beginning of the string/array area (pointed to by locations 140 and 141) and the new address to which you want the string to point. AH = INT(A/256):AL = A-(AH*256). This converts the offset we calculated above into the high byte AH and the low byte AL.

VVT = PEEK(134) + PEEK(135)*256. This poins the variable VVT to the beginning of the Variable Value Table.

Q = VVT + VNUM*8 + 2. This points the variable Q to the third byte (where the string/array offset is contained) of the VVT entry for the specific string to be modified. Since each table entry is 8 bytes long we multiply the variable number by eight to get the exact offset into the table. Don't forget the first variable number is zero, not one. The +2 in the statement points Q to the third byte of the entry.

The next two lines of the string magic routine are not really required and can be taken out after the program is debugged and working correctly. The lines are simply checks to insure the validity of the ADDR parameter and the VNUM parameter. The ADDR must point past the beginning of the string/array area. The VNUM must point to a string variable.

POKE Q,AL:POKE Q + 1,AH. This changes the string/array offset (bytes 3 and 4 of the entry) to the new offset. This will cause the string to point to our new address. Whenever you reference a string, BASIC gets the memory address of the string by adding the offset contained in the VVT to the address of the string/array area. Since the value we just poked in contains the difference between the beginning of the string/array area and the address to which we want to point the string, when the string is referenced we will point to the new address.

POKE Q + 2,AL:POKE Q + 3,AH. This changes the current size of the string to the value contained in the SIZE parameter. The current string size is stored in positions 5 and 6 of the VVT entry.

POKE Q + 4,AL:POKE Q + 5,AH:RETURN. This changes the dimensioned size of the string to the value contained in the SIZE parameter. The dimensioned size of the string is stored in positions 7 and 8 of the VVT entry. That's all there is to it. Now you know how to use the string magic routine. I think you will be surprised at the things you can do with it.

Table 1

Variable Value Table entry for strings

(data is listed by position, followed by the contents)

1. 128 indicates an undimensioned string. 129 indicates a dimensioned string.
2. Not used for strings.
3, 4. Offset into the string/array area — low byte first.
5, 6. Current string length — low byte first.
7, 8. Dimensioned string length — low byte first.

— Charlie Parker

RAMTALKER

(reprint: STATUS, January, 1985)

This is the first in a series of articles on the digital recording of sound with your Atari. I will present several modifications to what was originally an awkwardly coded and hard to follow program first appearing in BYTE magazine and adapted from that Apple program in the July, 1983 issue of ANTIC. In the course of updating this program, we should all learn something about programming, digital sound and special functions of the Atari. Rather than enumerate the changes made to Ed Stewart's program (that program is now almost unrecognizable), I will discuss the specific points of the program as it now exists.

The program requires a special circuit to allow the Atari to read an analog voltage at Port 3 (sorry, XL owners, we'll fix this problem next month; in the meantime, get your circuit built). The schematic is included in this article. I have found the 2 MegOhm potentiometer included in the original circuit schematic may be eliminated with no ill effects on its operation. Once you have the circuit built (see accompanying construction article), you may read a changing voltage (as might be produced by a microphone), as a resistance value from 0 to 255. The circuit works well in its present form, although we may change it in future articles.

The program, RAMTALKER, is a friendly, fast, easy to use program. After initialization, a menu is presented. To perform a desired function, press the number corresponding to the function. A BASIC GET command eliminates the need to type Return. If the function you pick is not the one you want, press Return and the program will take you back to the original menu. After a function is selected, the program will prompt you for more information.

RECORD asks for a sample speed. Sample speed is the speed at which the program will read the information coming in at the port. A sample speed of 1 will render the highest quality sound, while a value of 255 will result in nearly unintelligible noise. Once a sample speed is specified, followed by a Return, press the Start key to begin recording.

PLAY asks for a sample speed. This will be the speed at which the sound information contained in memory will be played back. A good speed is usually around 55, giving a natural sound to the playback. Of course, you may wish to have your recorded sounds resemble the Chipmunks or Lurch, in which case you will choose a higher or lower speed. Again, pressing Start after giving a sample speed will begin the playback.

THROUGHPUT asks for a recording sample speed, and will allow you to play sounds through the speaker with no time limitations. Press Start to begin, and a System Reset will get you out of this one.

SAVE and LOAD ask you for a file name. Include "D:, of "C: in the file specification. The program uses Atari's Central Input/Output (CIO) routines, which makes saving and loading sound files quite fast, even though sound files are 132 sectors long (single density).

WAVEFORM PLOT does just what it says. It plots a picture of the sound stored in memory (in locations 16384 to 32767, a full 16k) on a graph of Time against Frequency. The Time at a sample speed of 1 is a little over 7 seconds. I have not measured the frequency response of the system. We'll do that in another article. The sound is divided into 4 separate bands, so we are able to plot the entire contents of memory with some detail. To me, the waveform plotting routine is an exciting feature of this program. You can say a few words into your Atari, and then have the computer show you what your voice looks like. You can see how different sounds are similar, and where they are different. Looking at a plot of myself saying "file" and "while" gives me quite an appreciation for the difficult task a speech recognition system has to perform.

These are the basics of RAMTALKER. In the coming months we will modify the program even further. Some things I hope we can do with the program: editing the sounds in memory; improvement of sound quality, special effects (echo and speed effects); and maybe, just maybe, some speech regognition.

I am sure you have your own ideas as to where this program could go. I will be happy to hear your suggestions, criticism and comments. Next month will be a quick intro to digital sound theory, a new waveform plotting routine, and a few esoteric modifications.

— Randy Holmes

RAMTALKER CIRCUIT

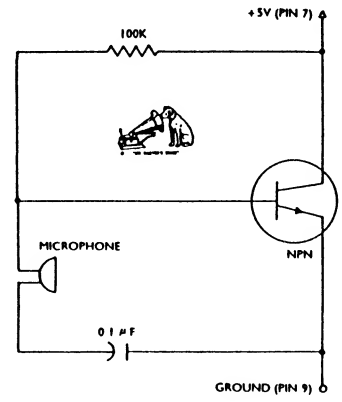
Construction Notes and Parts List

In place of a microphone, you may want to substitute a simple quarter-inch jack or RCA jack to allow you to plug in a guitar, keyboard, or tape player. This will give you higher quality sound than recording from a mic will. Adjust the volume control on your sound source to get the best, most distortion-free sound.

Parts: 1) .1 uF nonpolarized capacitor, available at Radio Shack. 2) NPN transistor 2N2222. This is a general purpose transistor, almost any NPN will work. 3) 100kOhm fixed resistor (brown-black-yellow resistor code). 4) DB-9 connector plug (joystick plug to you and me).

You may mount these components on a small circuit board (see Figure 2) or simply wire them together without a board; either way will work fine. Remember not to keep the soldering iron on the transistor too long, or you may damage the component. The same goes for the resistor and the capacitor, but they have a higher tolerance for heat. Also, be sure to observe the Emitter-Base-Collector specifications in the circuit. The back of the transistor package should have the pins specified in a diagram much like the transistor in Figure 1.

After the circuit is constructed, simply run a two-conductor wire from the specified points on the circuit to the correct pins (7 & 9) on your port plug. Test the circuit by plugging it into Port 3, and run the RAMTALKER program. Select the THROUGHPUT option, with a sample speed of 1. Plug in a microphone, guitar, tape player or some other sound source, and see if any sound comes from the TV/Monitor speaker. If not, go back and check your wiring, making sure all connections are good. With this circuit up and running, you are ready to begin digital sound recording with your Atari computer!



Ralph Walden

Teaches Assembly Language— #4

For those of you using MAC/65, here are three macros which will display text on the screen. Before you use the macro "DISPLAY" or "SCREEN" you MUST have first used the macro MESSAGE. Once MESSAGE has been used, it can be referred to as a subroutine (as well as PUTEDIT). SCREEN will send 1 to 4 bytes to the screen. Example: SCREEN '1,32,155' this will send I SPACE RETURN to the screen. After 3 or four bytes, it's more efficient to use DISPLAY. DISPLAY will send whatever is in quotes plus on optional second byte parameter to the screen. It will NOT send a 0 (a graphics heart). Here are some examples of its use:

DISPLAY "Hello World!",155 This will display the message in quotes and move the cursor to the next line.

DISPLAY "What is the date?" This will display the message in quotes, and leave the cursor at the end of the message.

You can also call MESSAGE directly as a subroutine (after you have used it as a macro) by loading X with the high byte of the address of the text, A with the low byte of the address of the text, and a JSR MESSAGE.

```
0100      .MACRO MESSAGE
0110 MESSAGE STA $E0
0120      STX $E1
0130 @MESLOOP LDY #0
0140      LDA ($E0)
0150      BEQ @MESEND
0160      JSR PUTEDIT
0170      INC $E0
0180      BNE @MESLOOP
0190      INC $E1
0200      BNE @MESLOOP
0210 @MESEND RTS
0220 PUTEDIT TAY
0230      LDA $E407
0240      PHA
0250      LDA $E406
0260      PHA
0270      TYA
0280      RTS
0290      .ENDM
0300      .MACRO DISPLAY
0310      LDA #  @TEXT
0320      LDX # >@TEXT
0330      JSR MESSAGE
0340      JMP @OUT
0350 @TEXT .BYTE %$1
0360        .IF %0>1
0370        .BYTE %2
0380        .ENDIF
0390      .BYTE 0
0400 @OUT
0410      .ENDM
0420      .MACRO SCREEN
0430      LDA #%1
0440      JSR PUTEDIT
0450        .IF %0>1
0460        LDA #%2
0470        JSR PUTEDIT
0480        .ENDIF
0490        .IF %0>2
0500        LDA #%3
0510        JSR PUTEDIT
0520        .ENDIF
0530        .IF %0>3
0540        LDA #%4
0550        JSR PUTEDIT
0560        .ENDIF
0570        .IF %0>4
0580        .ERROR "TOO MANY ARGUMENTS IN SCREEN"
0590        .ENDIF
0600      .ENDM
```

BOUNTY BOB STRIKES BACK

As a member of A.C.E. I am writing to all of you out there in "Bounty Bob" land. This review is of the exciting new game of "Bounty Bobs". As the old adage goes, this game is worth every penny. "Bounty Bob Strikes Back" ($50, Big 5 Software, Box 9078-185, Van Nuys, CA 91409) is a sequel to "Miner 2049er", and is by far the best I have ever seen! What designer Bill Hogue has done this time really makes your hair stand on end — or, I should say, Bounty Bob's hair.

This new game is as impressive as a new car! The tremendous graphics will test your color monitor — that's for sure. There is 4 channel sound and even program flexibility. That's right, I said flexibility. You can change several game parameters while playing or before startup. You can change (are you ready for this): number of lives, number of players, number of joysticks. You can enable or disable the Secret Messages scattered throughout the caverns. You can change the Bonus Limits at which you get another player. You can change the difficulty: Easy - filled in framework stays filled and dead Mutants remain dead from one life to the next; Medium - Mutants move slightly faster; Hard - filled in framwork resets and dead Mutants resurrect. FHEW! Oops not finished yet!! The final skill (should I say Insane) level is called "C'MON". The bonus timer starts with 1000 fewer points. We aren't through yet. You can also enable or disable the pause key function. You can change the volume of the background music, time the high score display is seen, time the main screen is seen, and you can change "Yukon Yohan" (the evil deed doer for whom Bob searches) to smoker or gum chewer! What this means I can only guess. You can change the number of letters displayed on the high score screen. Now. You can also change the Special Code used by the programmers which is actuated by pressing the Option key. What this does is still unknown to me.

This gives you kind of an idea what is in store for good old Bounty Bob. The game resides in a 40k cartridge. There are 25 caverns and a lot of special equipment for Bob to use. Bob's task is still to secure all levels of the mine, but he may also capture Yukon Yohan. The Mutants have gotten plum nasty. In fact they have multiplied! There's up to 26 of the little buggers on one of the levels. Are you shakin' in your boots yet? By the way, be on the lookout for Mutants who can go up or down ladders, and even down slides! It seems as though they got smarter or something.

But Bounty Bob can also do some neat new tricks! He can jump short or long distances. And if he eats a Super Energy Food Bar, these little goodies will pep old Bob up for those extra long leaps and bounds. They do wear off in a short time.

Added features include Grain Elevators, Gravity Lift, Hydraulic Lifts, Suction Tubes, Mobile Suction Unit and Acid Rain. There may be more, but I haven't seen them yet. With all this stuff to keep you going you will have fun with this game.

PS: Before I forget, remember the phone number trick in Miner 2049er? Well, it is possible to be trapped in an area where not even death is possible. Use the phone number to get you "special help" to get out of the situation (this only works on certain screens).

— Stephen E. Warn
East Helena, MT

****

Notes from Stan Ockers

XYGraph

To use with the "APEFACE" interface, set switches to 1. No paper detect (down), 2. Print at CR (down), 3. 7 bits (up), and 4. Automatic Linefeed (up).

Labels

To print only a few envelopes if you don't want to bother with labels and have a friction-feed printer, adjust the length of the blank strings in lines 406 and 412.

A real improvment in the program would be a list printing addition as described in the expansion part of the text for "Labels". Hopefully ready for next month...

-Stan

Atari Computer Enthusiasts

A.C.E. is an independent, non-profit and tax exempt computer club and user's group with no connection to the Atari Company, a division of Warner Communication Company. We are a group interested in educating our members in the use of the Atari Computer and in giving the latest News, Reviews and Rumors.

All our articles, reviews and programs come from you, our members.

Our membership is world-wide; membership fees include the A.C.E. Newsletter. Dues are $14 a year for U.S., and $24 a year Overseas Airmail and include about 10 issues a year of the ACE Newsletter.
Subscription Dep't: 3662 Vine Maple Dr., Eugene, OR 97405.

**President— Robert Browning, 90 W. Myoak Dr., Eugene, OR 97404 503-689-1513

Vice Pres— Larry Gold, 1927 McLean Blvd., Eugene, OR 97405 503-686-1490

Secretary— Bruce Ebling, 1501 River Loop #1, Eugene, OR 97404 503-688-6872

Librarian— Ron and Aaron Ness, 374 Blackfoot, Eugene, OR 97404 503-689-7106.

Editors—
Mike Dunn, 3662 Vine Maple Dr., Eugene, OR 97405 503-344-6193
— Jim Bumpas, 4405 Dillard Rd., Eugene, OR 97405 503-484-9925
E.R.A.C.E. (Education SIG Editor)— Nora Young, 105 Hansen Lane, Eugene, OR 97404 503-688-1458

Send 50c stamps or coin ($1 overseas) to the Ness' for the new, updated ACE Library List-new in May 84 !

Best of ACE books

Volume 1 contains bound issues of the ACE Newsletter from the first issue, Oct 81 to June of 1982

Volume 2 covers July 1982 to June 1983

Only $12 each ($2 extra for Airmail). Available only from:

George Suetsugu
45-602 Apuapu St
Kanoehe, HI 96744

SortFinder 1.2
A composite index of Atari related articles from 5 popular computer periodicals from Apr '81 to June '83, including ACE. Only $6 for ACE member from:
Jim Carr, Valley Soft
2660 S.W. DeArmond
Corvallis, OR 97333

Bulletin Board
(503) 343-4352

On line 24 hours a day, except for servicing and updating. Consists of a Tara equipped 48K Atari 400 with a TARA keyboard, 2 double-density double sided disk drives with an ATR 8000 interface, 2- 8" double density disk drives, an Epson MX80 printer, a Hayes SmartModem; running the ARMUDIC Bulletin Board software written by Frank L. Huband, 1206 N. Stafford St., Arlington, VA 22201. See the Nov '82 issue for complete details.

ATARI COMPUTER ENTHUSIASTS
3662 Vine Maple Dr. Eugene OR 97405

FIRST CLASS MAIL